

[illegible]


```
1 0001 0 MODULE setterm ( IDENT = 'V04-000',
2 0002 0 ADDRESSING_MODE (EXTERNAL = GENERAL)) =
3 0003 1 BEGIN
4 0004 1
5 0005 1 *****
6 0006 1 *
7 0007 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
8 0008 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
9 0009 1 * ALL RIGHTS RESERVED.
10 0010 1 *
11 0011 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
12 0012 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
13 0013 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
14 0014 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
15 0015 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
16 0016 1 * TRANSFERRED.
17 0017 1 *
18 0018 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
19 0019 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
20 0020 1 * CORPORATION.
21 0021 1 *
22 0022 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
23 0023 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
24 0024 1 *
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1 ++
29 0029 1 FACILITY: SET Command
30 0030 1
31 0031 1 ABSTRACT:
32 0032 1
33 0033 1 This module implements the DCL command SET TERMINAL.
34 0034 1
35 0035 1 ENVIRONMENT:
36 0036 1
37 0037 1 VAX/VMS operating system, user mode
38 0038 1
39 0039 1 AUTHOR: Gerry Smith 21-Mar-1983
40 0040 1
41 0041 1
42 0042 1 Modified by:
43 0043 1
44 0044 1 V03-016 BLS0347 Benn Schreiber 29-AUG-1984
45 0045 1 Only activate smg if we really need to.
46 0046 1
47 0047 1 V03-015 EMB0112 Ellen M. Batbouta 2-AUG-1984
48 0048 1 Place the IOSM_NOECHO modifier on the readprompt
49 0049 1 QIO in module, INQUIRE_TYPE. This prevents the
50 0050 1 escape sequence for the VT200 terminals being displayed
51 0051 1 on the screen on a SET TERMINAL/INQUIRE command.
52 0052 1
53 0053 1 V03-014 JRL0003 John R. Lawson, Jr. 18-May-1984 15:18
54 0054 1 For reasons of speed, do not let the linker resolve
55 0055 1 external references to the SMG (shareable image).
56 0056 1 SMG is only required when SET TERMINAL is issued, not
57 0057 1 for any other SET command. Therefore, at the risk of
```


58	0058	1	slowing down SET TERMINAL, other SET's are sped up by
59	0059	1	linking to and activating SMG only when necessary.
60	0060	1	
61	0061	1	V03-013 EMD0083 Ellen M. Dusseault 12-Apr-1984
62	0062	1	If /protocol is specified in the command string,
63	0063	1	don't issue the warning message, /perm qualifier
64	0064	1	was not specified.
65	0065	1	
66	0066	1	V03-012 STAN3012 Stanley Rabinowitz 25-Mar-1984
67	0067	1	Allow "foreign" terminal capabilities LFFILL, CRFILL and FRAME
68	0068	1	to set the corresponding terminal characteristics.
69	0069	1	
70	0070	1	V03-011 EMD0068 Ellen M. Dusseault 13-Mar-1984
71	0071	1	Add warning message to tell the user that the
72	0072	1	permanent characteristics will be used since
73	0073	1	the command issued did not contain the /perm
74	0074	1	qualifier and the user does not have a channel
75	0075	1	assigned to the terminal which was specified in
76	0076	1	the command string.
77	0077	1	
78	0078	1	V03-010 EMD0061 Ellen M. Dusseault 8-Mar-1984
79	0079	1	Check to see if the terminal characteristic, regis,
80	0080	1	is available if the terminal is a pro or rainbow when
81	0081	1	processing the command set term/inq.
82	0082	1	
83	0083	1	V03-009 STAN009 Stanley Rabinowitz 4-Mar-1984
84	0084	1	Add support for "foreign" terminals.
85	0085	1	If the terminal device type is not known,
86	0086	1	call the RTL TERMTABLE interface routines
87	0087	1	to see if it is defined there. If it is,
88	0088	1	then set up the terminal characteristics from
89	0089	1	the TERMTABLE information.
90	0090	1	----
91	0091	1	Give symbolic name, data_bufsiz, to the size of the data_buffer.
92	0092	1	
93	0093	1	V03-008 EMD0051 Ellen M. Dusseault 28-Feb-1984
94	0094	1	Add support for SET TERMINAL/[NO]DEC CRT=(1,2) to
95	0095	1	implement new terminal characteristic, DEC CRT2.
96	0096	1	Also a new device name, PRO_SERIES is introduced.
97	0097	1	
98	0098	1	V03-007 MMD0234 Meg Dumont, 4-Feb-1984 14:42
99	0099	1	Add support for SET TERMINAL/PROTOCOL for switching terminal
100	0100	1	ports to and from asynch ddcmp decnet lines.
101	0101	1	
102	0102	1	V03-006 MIR0300 MICHAEL I. ROSENBLUM 9-JAN-1984
103	0103	1	Fix problems that were encountered during field test
104	0104	1	1. SET TER/LOCAL should set NOECHO
105	0105	1	2. SET TER/SPEED should clear autobaud
106	0106	1	3. SET TER/INQUIRE should send out the normal ansi sequence
107	0107	1	in seven bit mode as well as 8 bit mode.
108	0108	1	4. SET TER/LOG SHOULD REPORT THE SPEED CORRECTLY
109	0109	1	
110	0110	1	V03-005 MIR0083 Michael I. Rosenblum 23-Aug-1983
111	0111	1	Reset terminal characteristics on the terminal line if
112	0112	1	the set term/inq fails so not to leave the terminal in
113	0113	1	8-bit mode.
114	0114	1	

SETTERM
V04-000

C 10
16-Sep-1984 01:10:06 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:09:20 [CLIUTL.SRC]SETTERM.B32;1

Page 3
(1)

: 115 0115 1 :
: 116 0116 1 :
: 117 0117 1 :
: 118 0118 1 :
: 119 0119 1 :
: 120 0120 1 :
: 121 0121 1 :
: 122 0122 1 :
: 123 0123 1 :
: 124 0124 1 :
: 125 0125 1 :
: 126 0126 1 :
: 127 0127 1 :
: 128 0128 1 :
: 129 0129 1 :
: 130 0130 1 :
: 131 0131 1 :--

V03-004 MIR0071 Michael I. Rosenblum 22-Jul-1983
Add code to inquire to parse 8-Bit inquire sequences.
add code to timeout all writes this will some timeing
problems when the terminal is control-s'ed

V03-003 GAS0143 Gerry Smith 22-Jun-1983
Lower all non-syntax ERRORS to WARNINGS.

V03-002 MIR0035 Michael I. Rosebnblum 27-Apr-1983
Add support for VT200 series terminals and VT200
Inquire sequences. Add support for multiple frame
sizes, Dismissing parity errors. Fix /Log qualifier.

V03-001 GAS0117 Gerry Smith 7-Apr-1983
If a terminal is set /AUTOBAUD, also set the speed to 9600.


```
133 0132 1 |
134 0133 1 | Include files
135 0134 1 |
136 0135 1 | LIBRARY 'SYSS$LIBRARY:LIB';           ! VAX/VMS common definitions
137 0136 1 | REQUIRE 'SRC$:SHOWDEF';             ! SHOW common definitions
138 0235 1 |
139 0236 1 |
140 0237 1 | Define macros for the fields within each terminal block. These
141 0238 1 | also correspond to the fields in the SENSEMODE/SENSECHAR block.
142 0239 1 |
143 0240 1 | MACRO
144 0241 1 |     term$b_class = 0, 0, 8, 0%,
145 0242 1 |     term$b_type  = 1, 0, 8, 0%,
146 0243 1 |     term$w_width = 2, 0, 16, 0%,
147 0244 1 |     term$b_page  = 7, 0, 8, 0%,
148 0245 1 |     term$l_set1  = 4, 0, 32, 0%,
149 0246 1 |     term$l_set2  = 8, 0, 32, 0%,
150 0247 1 |     term$l_clr1  = 12, 0, 32, 0%,
151 0248 1 |     term$l_clr2  = 16, 0, 32, 0%,
152 0249 1 |     term$l_rspnum = 20, 0, 32, 0%,
153 0250 1 |     term$l_rspblk = 24, 0, 32, 0%;
154 0251 1 |
155 0252 1 |
156 0253 1 | The data that gets used by all the subroutines is more easily handled
157 0254 1 | in Bliss if it is actually a vector of data. Then, thru a BIND, all
158 0255 1 | the separate names can be used to identify the various elements of the
159 0256 1 | vector. The length of the vector (in longwords) is given by the
160 0257 1 | literal data_bufsiz.
161 0258 1 |
162 M 0259 1 | MACRO bind_data =
163 M 0260 1 |     BIND
164 M 0261 1 |         tt1_set    = data_buffer[0] : BITVECTOR[32],
165 M 0262 1 |         tt1_clr    = data_buffer[1] : BITVECTOR[32],
166 M 0263 1 |         tt2_set    = data_buffer[2] : BITVECTOR[32],
167 M 0264 1 |         tt2_clr    = data_buffer[3] : BITVECTOR[32],
168 M 0265 1 |         speed      = data_buffer[4],
169 M 0266 1 |         parity     = data_buffer[5],
170 M 0267 1 |         fill       = data_buffer[6],
171 M 0268 1 |         flags      = data_buffer[7] : $BBLOCK[4],
172 M 0269 1 |         dev_desc   = data_buffer[8] : $BBLOCK[dsc$c_s_bln],
173 M 0270 1 |         info_block = data_buffer[10] : $BBLOCK[12],
174 M 0271 1 |         index      = data_buffer[13],
175 M 0272 1 |         chan       = data_buffer[14] : WORD,
176 M 0273 1 |         deccrt_set = data_buffer[15] : BITVECTOR[32],
177 M 0274 1 |         deccrt_clr = data_buffer[16] : BITVECTOR[32],
178 0275 1 |         name_desc  = data_buffer[17] : $BBLOCK[dsc$c_s_bln]; %;
179 0276 1 |
180 0277 1 | LITERAL
181 0278 1 |
182 0279 1 |     data_bufsiz = 19;           ! Number of longwords in data_buffer.
183 0280 1 |
184 0281 1 |
185 0282 1 | Define FLAGS bits
186 0283 1 |
187 0284 1 | MACRO
188 0285 1 |     set$v_lng  = 0, 0, 1, 0%,           ! Logging desired
189 0286 1 |     set$v_perm = 0, 1, 1, 0%,           ! Permanent chars requested
```


SETTERM
V04-000

E 10
16-Sep-1984 01:10:06 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:09:20 [CLIUTL.SRC]SETTERM.B32;1

Page 5
(2)

```
: 190      0287 1  set$y_odd   = 0, 2, 1, 0%,
: 191      0288 1  set$y_even  = 0, 3, 1, 0%,
: 192      0289 1  set$y_nopar = 0, 4, 1, 0%,
: 193      0290 1  set$y_lf    = 0, 5, 1, 0%,
: 194      0291 1  set$y_cr    = 0, 6, 1, 0%,
: 195      0292 1  set$y_speed = 0, 7, 1, 0%,
: 196      0293 1  set$y_width = 0, 8, 1, 0%,
: 197      0294 1  set$y_page  = 0, 9, 1, 0%,
: 198      0295 1  set$y_frame = 0,10, 1, 0%,
: 199      0296 1  set$y_dismis= 0,11, 1, 0%,
: 200      0297 1  set$y_nodism= 0,12, 1, 0%,
: 201      0298 1  set$y_vt200 = 0,13, 1, 0%,
: 202      0299 1  set$y_network=0,14, 1, 0%,
: 203      0300 1
```

```
: Odd parity requested
: Even parity requested
: Parity turned off
: LFfill given
: CRfill given
: New speed given
: Width changed
: Page length changed
: frame size changed
: Dismiss parityerror
: Dismiss parity turned off
: vt200 series terminals
: Set to set port to a decnet port
```

```
205 0301 1 |
206 0302 1 | Table of contents
207 0303 1 |
208 0304 1 |
209 0305 1 FORWARD ROUTINE
210 0306 1   set$terminal : NOVALUE,
211 0307 1   write_timeout : NOVALUE,
212 0308 1   get_term_type,
213 0309 1   get_term_def,
214 0310 1   inquire_type,
215 0311 1   get_values,
216 0312 1   log_results : NOVALUE;
217 0313 1 |
218 0314 1 |
219 0315 1 | External routines
220 0316 1 |
221 0317 1 EXTERNAL ROUTINE
222 0318 1 |   smg$init_term_table,
223 0319 1 |
224 0320 1 |
225 0321 1 |   smg$get_term_data,
226 0322 1 |   smg$del_term_table,
227 0323 1 |
228 0324 1 |   switch_to_terminal,
229 0325 1 |   switch_to_ddcmp,
230 0326 1 |   str$append,
231 0327 1 |   lib$cvt_dtb,
232 0328 1 |   cli$get_value,
233 0329 1 |   cli$present;
234 0330 1 |
235 0331 1 |
236 0332 1 |
237 0333 1 | Mechanisms for Post-activation linking to SMG
238 0334 1 |
239 0335 1 |
240 0336 1 macro   ! So I don't need to change every occurrence in the program
241 0337 1
242 0338 1   SMG$INIT_TERM_TABLE = ( . $SMG$INIT_TERM_TABLE ) %,
243 0339 1   SMG$DEL_TERM_TABLE = ( . $SMG$INIT_TERM_TABLE ) %,
244 0340 1   SMG$GET_TERM_DATA = ( . $SMG$GET_TERM_DATA ) %;
245 0341 1
246 0342 1 external routine   ! To do the actual linking
247 0343 1
248 0344 1   LIB$FIND_IMAGE_SYMBOL;
249 0345 1
250 0346 1 own   ! To hold the addresses of the actual routines
251 0347 1
252 0348 1   $SMG$INIT_TERM_TABLE,
253 0349 1   $SMG$DEL_TERM_TABLE,
254 0350 1   $SMG$GET_TERM_DATA: long;
255 0351 1
256 0352 1 |
257 0353 1 |
258 0354 1 | External references
259 0355 1 |
260 0356 1 EXTERNAL
261 0357 1   protocol$_none : vector,   ! Protocol type "NONE" descriptor
```



```
262 0358 1 protocol$ddcmp : vector,      ! Protocol type "DDCMP" descriptor
263 0359 1 term$tablc : BLOCKVECTOR[,28,BYTE], ! Table of known terminals
264 0360 1 term$name : VECTOR,          ! Table of their names
265 0361 1 term$ttset_key : VECTOR,     ! Keywords
266 0362 1 term$ttset_bit : VECTOR,     ! and their bitmasks (devdepend)
267 0363 1 term$ttclr_key : VECTOR,     ! Keywords (inverse)
268 0364 1 term$ttclr_bit : VECTOR,     ! and their bitmasks
269 0365 1 term$tt2set_key : VECTOR,    ! Keywords and
270 0366 1 term$tt2set_bit : VECTOR,    ! their bitmasks (devdepnd2)
271 0367 1 term$tt2clr_key : VECTOR,    ! Inverted keywords
272 0368 1 term$tt2clr_bit : VECTOR,    ! their bits
273 0369 1 term$reqblk : VECTOR,        ! Vector of request strings
274 0370 1 term$odd : VECTOR,           ! "Odd" descriptor
275 0371 1 term$even : VECTOR,          ! "Even" descriptor
276 0372 1 term$none : VECTOR,          ! "None" descriptor
277 0373 1 term$spdbl : VECTOR;         ! Vector of speed descriptors
278 0374 1
279 0375 1
280 0376 1
281 0377 1 ! Macro to generate descriptors names SD_string
282 0378 1
283 0379 1 MACRO
284 M 0380 1 SD[A] =
285 0381 1 BIND %NAME('SD_',A) = $DESCRIPTOR(A)%;
286 0382 1
287 P 0383 1 SD(
288 P 0384 1 'SMGSHR',
289 P 0385 1 'PROTOCOL',
290 P 0386 1 'PARITY',
291 P 0387 1 'FRAME',
292 P 0388 1 'PAGE',
293 P 0389 1 'WIDTH',
294 P 0390 1 'CRFILL',
295 P 0391 1 'LFFILL',
296 P 0392 1 'SPEED',
297 P 0393 1 'DEC_CRT',
298 P 0394 1 'NO',
299 P 0395 1 'DEVICE TYPE',
300 0396 1 'DEC_CRT2');
301 0397 1
302 0398 1 BIND
303 0399 1 SD_COMMA = $DESCRIPTOR(','); !Descriptor of a comma
304 0400 1
305 0401 1 ! Declare some shared messages
306 0402 1
307 P 0403 1 $SHR_MSGDEF (SET,119,LOCAL,
308 0404 1 (invquaval, error));
309 0405 1
310 0406 1
311 0407 1
312 0408 1 ! Declare literals defined elsewhere
313 0409 1
314 0410 1 EXTERNAL LITERAL
315 0411 1 term$num, ! Number of known terminals
316 0412 1 term$ttset_num, ! Number of DEVDEPEND bits/keywords
317 0413 1 term$ttclr_num, ! Number of inverted bits/keywords
318 0414 1 term$tt2set_num, ! Number of DEVDEPND2 bits/keywords
```

SETTERM
V04-000

H 10
16-Sep-1984 01:10:06 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:09:20 [CLIUTL.SRC]SETTERM.B32;1

Page 8
(3)

```
.. 319      0415 1    term$_tt2clr_num,  
.. 320      0416 1    term$_spdnum,  
.. 321      0417 1    term$_regnum,  
.. 322      0418 1    term$_vk100,  
.. 323      0419 1    term$_vt100,  
.. 324      0420 1    term$_vt101,  
.. 325      0421 1    term$_vt102,  
.. 326      0422 1    term$_vt105,  
.. 327      0423 1    term$_vt125,  
.. 328      0424 1    term$_vt131,  
.. 329      0425 1    term$_vt132,  
.. 330      0426 1    term$_vt173,  
.. 331      0427 1    term$_vt52,  
.. 332      0428 1    term$_vt200_series,  
.. 333      0429 1    term$_pro_series,  
.. 334      0430 1    term$_ft1,  
.. 335      0431 1    term$_ft8,  
.. 336      0432 1    term$_unknown,  
.. 337      0433 1    set$_termset,  
.. 338      0434 1    set$_writeerr,  
.. 339      0435 1    set$_unkterm,  
.. 340      0436 1    set$_noperm,  
.. 341      0437 1    cli$_ivdevtype,  
.. 342      0438 1    cli$_absent,  
.. 343      0439 1    cli$_negated,  
.. 344      0440 1    cli$_present;  
.. 345      0441 1
```

```
! Number of inverted bits/keywords  
! Number of terminal speeds  
! Number of request strings  
! Identifiers for the VT1XX family  
! of non-compatible terminals  
  
! old clunkers  
! Vt200 series terminals  
! Pro series terminals  
! Beginning of foreign terminals  
! End of foreign terminals  
! "Unknown terminal" type  
! Terminal characteristics set  
! Error modifying device  
! Terminal type unknown  
! Permanent qualifier not specified  
! Invalid device type  
! Qualifier absent  
! Qualifier explicitly negated  
! Qualifier explicitly present
```



```
0442 1 GLOBAL ROUTINE set$terminal : NOVALUE =
0443 BEGIN
0444
0445 !++
0446 Functional description
0447
0448 This is the routine for the SET TERMINAL command. It is called
0449 from the SET command processor, and sets the terminal characteristics.
0450
0451 Inputs
0452 None
0453
0454 Outputs
0455 None
0456
0457 ----
0458
0459 LOCAL
0460 status, ! Status return
0461 set_length, ! Address of change length string
0462 dev_char : $BBLOCK[4], ! Store the device char from GETDVI
0463 dev_buffer : VECTOR[20, BYTE], ! Device buffer
0464 data_buffer : VECTOR[data_bufsiz] ! Buffer to hold much data
0465 INITIAL (REP data_bufsiz OF (0)), ! initially clear
0466 info_desc : VECTOR[2], ! $GETCHN descriptor
0467 iosb : VECTOR[4, WORD], ! I/O status block
0468 default_device : long initial (0), ! flag set to 1 device not specified
0469 use default
0470 dvi_list2 : $ITMLST_DECL(ITEMS=1), ! item list for ref count
0471 refcount : long initial (0); ! reference count for terminal device
0472 BIND
0473 timeout = UPLIT(-5*10*1000*1000,-1); ! 5 seconds
0474
0475 !
0476 Bind all DATA_BUFFER to nice normal names
0477
0478 bind_data;
0479
0480 !
0481 Collect the name of the device.
0482
0483 $init_dyndesc(dev_desc); ! Make the descriptor dynamic
0484
0485 IF NOT cli$get_value(%ASCID 'DEVICE', ! Get the device name
0486 dev_desc)
0487 THEN
0488 BEGIN
0489 default_device = 1; ! Use default device
0490 dev_desc[dsc$w_length] = %CHARCOUNT('SYSS$COMMAND');
0491 dev_desc[dsc$a_pointer] = UPLIT BYTE('SYSS$COMMAND');
0492 END;
0493
0494 !
0495 Use GETDVI to determine the real device name.
0496
0497
0498 BEGIN
```

```
404      0499 LOCAL
405      0500     class,
406      0501     dvi_list : $ITMLST DECL(ITEMS = 3);
407      0502 $ITMLST_INIT(ITMLST = dvi_list,
408      0503     (ITMCOD = dvi$devclass,
409      0504     BUFADR = class),
410      0505     (ITMCOD = dvi$devchar,
411      0506     BUFADR = dev_char),
412      0507     (ITMCOD = dvi$devnam,
413      0508     BUFADR = dev_buffer,
414      0509     BUFSIZ = %ALLOCATION(dev_buffer),
415      0510     RETLEN = dev_desc));
416      P 0511 status = $GETDVIW(ITMLST = dvi_list,
417      P 0512     DEVNAM = dev_desc,
418      0513     IOSB = iosb);
419      0514 IF .status
420      0515 THEN status = .iosb[0];
421      0516 IF NOT .status
422      0517 THEN
423      0518 BEGIN
424      0519     SIGNAL(set$writeerr AND NOT sts$m_severity OR sts$k_warning,
425      0520     1, dev_desc, .status);
426      0521 RETURN;
427      0522 END
428      0523 ELSE
429      0524 BEGIN
430      0525     dev_desc[dsc$w_length] = .dev_desc[dsc$w_length] - 1;
431      0526     dev_desc[dsc$a_pointer] = dev_buffer + 1;
432      0527     IF .class NEQ dc$term
433      0528     THEN
434      0529 BEGIN
435      0530     SIGNAL(set$writeerr AND NOT sts$m_severity OR sts$k_warning,
436      0531     1, dev_desc,
437      0532     cli$_ivdevtype);
438      0533 RETURN;
439      0534 END;
440      0535 END;
441      0536 END;
442      0537
443      0538
444      0539 Assign a channel to the device.
445      0540
446      P 0541 IF NOT (status = $ASSIGN(DEVNAM = dev_desc,
447      0542     CHAN = chan))
448      0543 THEN
449      0544 BEGIN
450      0545     SIGNAL(set$writeerr AND NOT sts$m_severity OR sts$k_warning,
451      0546     1, dev_desc, .status);
452      0547 RETURN;
453      0548 END;
454      0549
455      0550
456      0551 ! If /PROTOCOL was specified then switch to the type of port requested.
457      0552 ! If a terminal port was requested =NONE then allow other terminal
458      0553 ! characteristics to be set. If Asynch DDCMP port was request then
459      0554 ! only allow the switch to made all other qualifiers are ignored. The
460      0555 ! user must have OPERATOR privilege in order to make the switch. The
```



```

: 461      0556 2 ! port being switched can not have a non-zero reference count and
: 462      0557 2 ! the Async DDCMP driver code must be loaded into the system.
: 463      0558 2
: 464      0559 2 IF cli$present(SD_PROTOCOL)
: 465      0560 2 THEN
: 466      0561 2     BEGIN
: 467      0562 2     LOCAL
: 468      0563 2         desc : $BBLOCK [dsc$c_s_bln],
: 469      0564 2         process_privs : $BBLOCK[8],
: 470      0565 2         getjpi_list : $ITMLST_DECL(ITEMS=1);
: 471      P 0566 2     $ITMLST_INIT(ITMLST=getjpi_list,
: 472      0567 2         (ITMCOD = jpi$procpriv,BUFADR=process_privs));
: 473      0568 2     $INIT_DYNDESC(desc);
: 474      0569 2
: 475      0570 2     ! Assume that the port isn't going to become a decnet port
: 476      0571 2
: 477      0572 2     flags[set$v_network] = 0;
: 478      0573 2
: 479      0574 2     status = $GETJPI(ITMLST=getjpi_list);
: 480      0575 2     IF NOT .status
: 481      0576 2     THEN
: 482      0577 2         BEGIN
: 483      0578 2         SIGNAL(set$writeerr AND NOT sts$m_severity OR sts$k_warning,
: 484      0579 2             T, dev_desc, .status);
: 485      0580 2         RETURN;
: 486      0581 2         END;
: 487      0582 2     IF NOT .process_privs[prv$v_oper]
: 488      0583 2     THEN
: 489      0584 2         BEGIN
: 490      0585 2         SIGNAL(set$writeerr AND NOT sts$m_severity OR sts$k_warning,
: 491      0586 2             T, dev_desc, ss$_nopriv);
: 492      0587 2         RETURN;
: 493      0588 2         END;
: 494      0589 2     IF cli$get_value(SD_PROTOCOL, desc)
: 495      0590 2     THEN
: 496      0591 2         BEGIN
: 497      0592 2         LOCAL arglist : vector [2];
: 498      0593 2         IF CH$EQL (.desc[dsc$w_length], .desc[dsc$a_pointer],
: 499      0594 2             .desc[dsc$w_length], .protocol$_none[1])
: 500      0595 2         THEN
: 501      0596 2             BEGIN
: 502      0597 2             arglist[0] = 1;
: 503      0598 2             arglist[1] = .chan;
: 504      P 0599 2             IF NOT (STATUS = $CMKRNL(ROUTIN = switch_to_terminal,
: 505      0600 2                 ARGST = arglist))
: 506      0601 2             THEN
: 507      0602 2                 BEGIN
: 508      0603 2                 SIGNAL(set$writeerr AND NOT sts$m_severity
: 509      0604 2                     OR sts$k_warning,1, dev_desc, .status);
: 510      0605 2                 RETURN;
: 511      0606 2                 END;
: 512      0607 2
: 513      0608 2             ! Clear the network bit in dev_char. Now that the device
: 514      0609 2             ! is a terminal again we can allow the remainder
: 515      0610 2             ! of the set to happen.
: 516      0611 2
: 517      0612 2         dev_char[dev$v_net] = 0;
```

```
.. 518 0613 5
.. 519 0614 4
.. 520 0615 5
.. 521 0616 5
.. 522 0617 5
.. 523 0618 5
.. 524 0619 5
.. 525 0620 5
.. 526 0621 5
.. 527 0622 5
.. 528 0623 5
.. 529 0624 6
.. 530 0625 6
.. 531 0626 6
.. 532 0627 5
.. 533 0628 6
.. 534 0629 6
.. 535 0630 6
.. 536 0631 6
.. 537 0632 6
.. 538 0633 6
.. 539 0634 6
.. 540 0635 5
.. 541 0636 4
.. 542 0637 3
.. 543 0638 4
.. 544 0639 4
.. 545 0640 4
.. 546 0641 4
.. 547 0642 4
.. 548 0643 2
.. 549 0644 2
.. 550 0645 2
.. 551 0646 2
.. 552 0647 2
.. 553 0648 2
.. 554 0649 2
.. 555 0650 2
.. 556 0651 2
.. 557 0652 2
.. 558 0653 2
.. 559 0654 2
.. 560 0655 2
.. 561 0656 2
.. 562 0657 2
.. 563 0658 2
.. 564 0659 2
.. 565 0660 2
.. 566 0661 2
.. 567 0662 2
.. 568 0663 2
.. 569 0664 2
.. 570 0665 2
.. 571 0666 2
.. 572 0667 2
.. 573 0668 2
.. 574 0669 2

      END
    ELSE
      BEGIN
        ! Switch to a decnet device only after the terminal
        ! characteristics have been updated to the ones
        ! the user specified on the command line.
        IF CH$EQL (.desc[dsc$w_length], .desc[dsc$a_pointer],
                  .desc[dsc$w_length], .protocol$_ddcmp[1])
        THEN
          BEGIN
            flags[set$v_network] = 1;
          END
        ELSE
          BEGIN
            SIGNAL(set$writeerr AND NOT sts$m_severity
                  OR sts$k_warning, 1, dev_desc,
                  set$invquaval,
                  2, desc, SD_PROTOCOL);
          END
          RETURN;
        END
      END
    END
  ELSE
    BEGIN
      SIGNAL(set$writeerr AND NOT sts$m_severity OR sts$k_warning,
            1, dev_desc, .status);
    END
    RETURN;
  END
END;

! If the NET bit is set in DEV_CHAR then this is a DECNET terminal port
! and allowing the set continue can produce strange results.
IF .dev_char[dev$v_net]
THEN
  BEGIN
    SIGNAL(set$writeerr AND NOT sts$m_severity OR sts$k_warning,
          1, dev_desc, ss$_termnetdev);
  END
  RETURN;
END;

! Determine if this is /PERM or just normal. See if logging is required.
flags[set$v_perm] = cli$present(%ASCID 'PERMANENT');
flags[set$v_log] = cli$present(%ASCID 'LOG');

! Generate a warning message under the following conditions:
! 1. a specific terminal device was specified,
! 2. the perm qualifier is absent from the command,
! 3. the refcount is 1 (this routine has the only channel),
! 4. the protocol qualifer is absent from the command
! The warning message will tell the user that the permanent characteristics
! will be used after this routine deassigns its channel to the terminal
! (ref count will drop to 0). Therefore it is possible that his efforts to
```



```
575 0670 2 ! modify characteristics will be fruitless .
576 0671 2
577 0672 2 $ITMLST_INIT( ITMLST = dvi_list2, ( ITMCOD = dvi$_refcnt, BUFADR = refcount ) ) ;
578 0673 2 status = $GETDVIW ( ITMLST = dvi_list2, DEVNAM = dev_desc, iosb = iosb ) ;
579 0674 2 IF .status
580 0675 2 THEN status = .iosb[0];
581 0676 2 IF NOT .status
582 0677 2 THEN
583 0678 2 BEGIN
584 0679 2     SIGNAL( set$_writeerr AND NOT sts$m_severity OR sts$k_warning,
585 0680 2         1, dev_desc, .status );
586 0681 2     RETURN;
587 0682 2 END
588 0683 2 ELSE
589 0684 2 BEGIN
590 0685 2     IF ( .refcount EQL 1 )
591 0686 2         AND
592 0687 2         ( NOT .default_device )
593 0688 2         AND
594 0689 2         ( NOT .flags[ set$v_perm ] )
595 0690 2         AND
596 0691 2         ( NOT cli$present( SD_PROTOCOL ) )
597 0692 2     THEN
598 0693 2         SIGNAL( set$_noperm AND NOT sts$m_severity OR sts$k_warning ) ;
599 0694 2     END;
600 0695 2 !
601 0696 2 ! Get the characteristics
602 0697 2 !
603 0698 2 status = $QIOW( CHAN = .chan,
P 0699 2     FUNC = ( IF .flags[ set$v_perm ]
P 0700 2         THEN io$_sensechar
P 0701 2         ELSE io$_sensemode ),
P 0702 2     IOSB = iosb,
P 0703 2     P1 = info_block,
0704 2     P2 = 12 );
610 0705 2 IF .status
611 0706 2 THEN status = .iosb[0];
612 0707 2 IF NOT .status
613 0708 2 THEN
614 0709 2 BEGIN
615 0710 2     SIGNAL( set$_writeerr AND NOT sts$m_severity OR sts$k_warning,
616 0711 2         1, dev_desc, .status );
617 0712 2     RETURN;
618 0713 2 END;
619 0714 2 !
620 0715 2 ! zero the parity flags and fill characteristics.
621 0716 2 !
622 0717 2 parity = 0;
623 0718 2 fill = 0;
624 0719 2 !
625 0720 2 !
626 0721 2 ! See if a specific terminal type was specified. If an error, then it has
627 0722 2 ! already been signaled, and we should just go away.
628 0723 2 !
629 0724 2 IF NOT get_term_type( data_buffer )
630 0725 2 THEN RETURN;
631 0726 2
```

```
632 0727 2 !
633 0728 2 ! Get individual qualifiers. If an error occurred, then it has already been
634 0729 2 ! signaled and we should just return.
635 0730 2 !
636 0731 2 IF NOT get values(data_buffer)
637 0732 2 THEN RETURN;
638 0733 2 IF .flags[set$v_width] ! If width changed and this
639 0734 2 AND .SBBLOCK[info_block[term$l_set2], tt2$v_deccrt] ! and it's a DEC crt
640 0735 2 THEN IF .info_block[term$w_width] GTR 80 ! If wide screen
641 0736 2 THEN ! set length to 132
642 0737 2 BEGIN ! and fix page length
643 0738 2 set length = UPLIT BYTE (27, '[?3h');
644 0739 2 IF NOT .SBBLOCK[info_block[term$l_set2], tt2$v_avo]
645 0740 2 THEN info_block[term$b_page] = 14;
646 0741 2 END
647 0742 2 ELSE ! If narrow, set
648 0743 2 BEGIN ! length to 80
649 0744 2 set length = UPLIT BYTE (27, '[?3l'); ! and adjust page
650 0745 2 IF NOT .SBBLOCK[info_block[term$l_set2], tt2$v_avo]
651 0746 2 THEN info_block[term$b_page] = 24;
652 0747 2 END;
653 0748 2
654 0749 2
655 0750 2 !
656 0751 2 ! Now set the stuff explicitly requested by the user
657 0752 2 !
658 P 0753 2 status = $QIOW(CHAN = .chan,
659 P 0754 2 FUNC = (IF .flags[set$v_perm]
660 P 0755 2 THEN io$_setchar
661 P 0756 2 ELSE io$_setmode),
662 P 0757 2 IOSB = iosb,
663 P 0758 2 P1 = info_block,
664 P 0759 2 P2 = 12,
665 P 0760 2 P3 = .speed,
666 P 0761 2 P4 = .fill,
667 0762 2 P5 = .parity);
668 0763 2 IF .status
669 0764 2 THEN status = .iosb[0];
670 0765 2 IF .status EQL ss$_incompat
671 0766 2 THEN SIGNAL(.status);
672 0767 2 IF NOT .status
673 0768 2 THEN
674 0769 2 Begin
675 0770 2 SIGNAL(set$_writeerr AND NOT sts$m_severity OR sts$k_warning,
676 0771 2 1, dev_desc, .status);
677 0772 2 RETURN;
678 0773 2 End;
679 0774 2
680 0775 2 ! Now check the flag and switch to a decnet port if the user requested to
681 0776 2 ! do so.
682 0777 2
683 0778 2 IF .flags[set$v_network]
684 0779 2 THEN
685 0780 2 BEGIN
686 0781 2 LOCAL arglist : vector[2];
687 0782 2 arglist[0] = 1;
688 0783 2 arglist[1] = .chan;
```



```
689 P 0784 4 IF NOT (STATUS = $CMKRNL(ROUTIN = switch_to_ddcmp,
690 0785 4 ARGST = arglist))
691 0786 4 THEN
692 0787 4 BEGIN
693 0788 4 SIGNAL(set$writeerr AND NOT sts$m_severity
694 0789 4 OR sts$k_warning, 1, dev_desc, .status);
695 0790 4 RETURN;
696 0791 4 END
697 0792 4 ELSE
698 0793 4 RETURN;
699 0794 4 END;
700 0795 4
701 0796 4 Now to set the characteristics. First check to see if, because the
702 0797 4 width has changed, we need to fix the screen. Then set the explicit stuff.
703 0798 4
704 0799 4 IF .flags[set$v_width] ! If width changed and this
705 0800 4 AND .SBBLOCK[info_block[term$l_set2], tt2$v_deccrt] ! and it's a DEC crt
706 0801 4 THEN
707 0802 4 BEGIN
708 0803 4
709 P 0804 4 $SETIMR(DAYTIM = timeout, ! Set timeout timer going
710 P 0805 4 ASTADR = write_timeout,
711 0806 4 REQIDT = .chan);
712 0807 4
713 P 0808 4 status = $QIOW(CHAN = .chan,
714 P 0809 4 FUNC = io$writevblk,
715 P 0810 4 P1 = .set_length,
716 P 0811 4 P2 = 5,
717 0812 4 IOSB = iosb);
718 0813 4
719 0814 4 $CANTIM(REQIDT = .chan); ! Cancel the timer
720 0815 4 IF .status
721 0816 4 THEN IF .iosb[0] NEQ 0 THEN status = .iosb[0];
722 0817 4 IF NOT .status
723 0818 4 THEN
724 0819 4 BEGIN
725 0820 4 SIGNAL(set$writeerr AND NOT sts$m_severity OR sts$k_warning,
726 0821 4 1, dev_desc, .status);
727 0822 4 RETURN;
728 0823 4 END;
729 0824 4 END;
730 0825 4 IF .SBBLOCK[info_block[term$l_set2], tt2$v_deccrt]
731 0826 4 OR (.info_block[term$b_type] EQL .term$ _table[term$_vt52, term$b_type])
732 0827 4 THEN
733 0828 4 BEGIN
734 0829 4 LOCAL set_kpstate;
735 0830 4
736 P 0831 4 $SETIMR(DAYTIM = timeout, ! Set timeout timer going
737 P 0832 4 ASTADR = write_timeout,
738 0833 4 REQIDT = .chan);
739 0834 4
740 P 0835 4 status = $QIOW(CHAN = .chan,
741 P 0836 4 FUNC = io$writevblk,
742 P 0837 4 P1 = (IF .SBBLOCK[info_block[term$l_set2], tt2$v_app_keypad]
743 P 0838 4 THEN UPLIT BYTE (27, '=') ELSE UPLIT BYTE (27, '>')),
744 P 0839 4 P2 = 2,
745 0840 4 IOSB = iosb);
```



```
746 0841 3 SCANTIM(REQIDT = .chan); ! Cancel the timer
747 0842 3 IF .status
748 0843 3 THEN IF .iosb[0] NEQ 0 THEN status = .iosb[0];
749 0844 3 IF NOT .status
750 0845 3 THEN
751 0846 3 BEGIN
752 0847 3 SIGNAL(set$writeerr AND NOT sts$m_severity OR sts$k_warning,
753 0848 3 1, dev_desc, .status);
754 0849 3 RETURN;
755 0850 3 END
756 0851 3 END;
757 0852 2 IF .flags[set$v_vt200]
758 0853 2 THEN
759 0854 2 BEGIN
760 0855 2 $SETIMR(DAYTIM = timeout, ! Set timeout timer going
761 P 0856 2 ASTADR = write_timeout,
762 P 0857 2 REQIDT = .chan);
763 0858 2
764 0859 2 status = $QIOW(CHAN = .chan,
765 P 0860 2 FUNC = io$writevblk,
766 P 0861 2 P1 = uplit byte (27, '[62'p', 27, ' F'),
767 P 0862 2 P2 = 9,
768 P 0863 2 IOSB = iosb);
769 0864 2
770 0865 2 SCANTIM(REQIDT = .chan); ! Cancel the timer
771 0866 2 END;
772 0867 2 IF .flags[set$v_log]
773 0868 2 THEN log_results(data_buffer);
774 0869 2
775 0870 2 RETURN;
776 0871 2 END;
777 0872 1
```

INFO#250

L1:0527

Referenced LOCAL symbol CLASS is probably not initialized

```
.TITLE SETTERM
.IDENT \V04-000\

.PSECT $SPLITS,NOWRT,NOEXE,2

52 48 53 47 4D 53 00000 P.AAB: .ASCII \SMGSHR\
00006 .BLKB 2
00000006 00008 P.AAA: .LONG 6
00000000 0000C .ADDRESS P.AAB
4C 4F 43 4F 54 4F 52 50 00010 P.AAD: .ASCII \PROTOCOL\
00000008 00018 P.AAC: .LONG 8
00000000 0001C .ADDRESS P.AAD
59 54 49 52 41 50 00020 P.AAF: .ASCII \PARITY\
00026 .BLKB 2
00000006 00028 P.AAE: .LONG 6
00000000 0002C .ADDRESS P.AAF
45 4D 41 52 46 00030 P.AAH: .ASCII \FRAME\
00035 .BLKB 3
00000005 00038 P.AAG: .LONG 5
00000000 0003C .ADDRESS P.AAH
45 47 41 50 00040 P.AAJ: .ASCII \PAGE\
```



```
00000004 00044 P.AAI: .LONG 4
00000000 00048 .ADDRESS P.AAJ
48 54 44 49 57 0004C P.AAL: .ASCII \WIDTH\
00051 .BLKB 3
00000005 00054 P.AAK: .LONG 5
00000000 00058 .ADDRESS P.AAL
4C 4C 49 46 52 43 0005C P.AAN: .ASCII \CRFILL\
00062 .BLKB 2
00000006 00064 P.AAM: .LONG 6
00000000 00068 .ADDRESS P.AAN
4C 4C 49 46 46 4C 0006C P.AAP: .ASCII \LFFILL\
00072 .BLKB 2
00000006 00074 P.AAO: .LONG 6
00000000 00078 .ADDRESS P.AAP
44 45 45 50 53 0007C P.AAR: .ASCII \SPEED\
00081 .BLKB 3
00000005 00084 P.AAQ: .LONG 5
00000000 00088 .ADDRESS P.AAR
54 52 43 5F 43 45 44 0008C P.AAT: .ASCII \DEC_CRT\
00093 .BLKB 1
00000007 00094 P.AAS: .LONG 7
00000000 00098 .ADDRESS P.AAT
4F 4E 0009C P.AAV: .ASCII \NO\
0009E .BLKB 2
00000002 000A0 P.AAU: .LONG 2
00000000 000A4 .ADDRESS P.AAV
45 50 59 54 5F 45 43 49 56 45 44 000AB P.AAX: .ASCII \DEVICE_TYPE\
000B3 .BLKB 1
0000000B 000B4 P.AAW: .LONG 11
00000000 000B8 .ADDRESS P.AAX
32 54 52 43 5F 43 45 44 000BC P.AAZ: .ASCII \DEC_CRT2\
00000008 000C4 P.AAY: .LONG 8
00000000 000C8 .ADDRESS P.AAZ
2C 000CC P.ABB: .ASCII \,\
000CD .BLKB 3
00000001 000D0 P.ABA: .LONG 1
00000000 000D4 .ADDRESS P.ABB
00000000# 000D8 P.ABC: .LONG 0[19]
00000000 F0050F80 00124 P.ABD: .LONG -50000000, -1
00 00 45 43 49 56 45 44 0012C P.ABF: .ASCII \DEVICE\<0><0>
010E0006 00134 P.ABE: .LONG 17694726
00000000 00138 .ADDRESS P.ABF
44 4E 41 4D 4D 4F 43 24 53 59 53 0013C P.ABG: .ASCII \SYSSCOMMAND\
00147 .BLKB 1
00 00 00 54 4E 45 4E 41 4D 52 45 50 00148 P.ABI: .ASCII \PERMANENT\<0><0><0>
010E0009 00154 P.ABH: .LONG 17694729
00000000 00158 .ADDRESS P.ABI
00 47 4F 4C 0015C P.ABK: .ASCII \LOG\<0>
010E0003 00160 P.ABJ: .LONG 17694723
00000000 00164 .ADDRESS P.ABK
1B 00168 P.ABL: .BYTE 27
68 33 3F 5B 00169 .ASCII \[?3h\
1B 0016D P.ABM: .BYTE 27
6C 33 3F 5B 0016E .ASCII \[?3l\
1B 00172 P.ABN: .BYTE 27
3D 00173 .ASCII \=\
1B 00174 P.ABO: .BYTE 27
```



```

70 22 32 36 58 18 46 20
3E 00175 .ASCII \>\
1B 00176 P.ABP: .BYTE 27
5B 00177 .ASCII \[62'p\
1B 0017C .BYTE 27
46 20 0017D .ASCII \ F\

.PSECT $OWNS,NOEXE,2

```

```

00000 $SMG$INIT TERM_TABLE:
      .BLKB 4
00004 $SMG$DEL TERM_TABLE:
      .BLKB 4
00008 $SMG$GET TERM_DATA:
      .BLKB 4

```

```

SD_SMGSHR= P.AAA
SD_PROTOCOL= P.AAC
SD_PARITY= P.AAE
SD_FRAME= P.AAG
SD_PAGE= P.AAI
SD_WIDTH= P.AAK
SD_CRFILL= P.AAM
SD_LFFILL= P.AAO
SD_SPEED= P.AAQ
SD_DEC_CRT= P.AAS
SD_NO= P.AAU
SD_DEVICE_TYPE= P.AAW
SD_DEC_CRT2= P.AAY
SD_COMMA= P.ABA
TIMEOUT= P.ABD

.EXTRN SWITCH_TO_TERMINAL
.EXTRN SWITCH_TO-DDCMP
.EXTRN STR$APPEND, LIB$CVT DTB
.EXTRN CLISGET VALUE, CLISPRESENT
.EXTRN LIB$FIND IMAGE_SYMBOL
.EXTRN PROTOCOL$ NONE, PROTOCOL$_DDCMP
.EXTRN TERMS_TABLE, TERMS_NAME
.EXTRN TERMS_TTSET_KEY
.EXTRN TERMS_TTSET_BIT
.EXTRN TERMS_TTCLR_KEY
.EXTRN TERMS_TTCLR_BIT
.EXTRN TERMS_TT2SET_KEY
.EXTRN TERMS_TT2SET_BIT
.EXTRN TERMS_TT2CLR_KEY
.EXTRN TERMS_TT2CLR_BIT
.EXTRN TERMS_REQBLK, TERMS_ODD
.EXTRN TERMS_EVEN, TERMS_NONE
.EXTRN TERMS_SPDBLK, TERMS_NUM
.EXTRN TERMS_TTSET_NUM
.EXTRN TERMS_TTCLR_NUM
.EXTRN TERMS_TT2SET_NUM
.EXTRN TERMS_TT2CLR_NUM
.EXTRN TERMS_SPDNUM, TERMS_REQNUM
.EXTRN TERMS_VK100, TERMS_VT100
.EXTRN TERMS_VT101, TERMS_VT102
.EXTRN TERMS_VT105, TERMS_VT125
.EXTRN TERMS_VT131, TERMS_VT132

```


				OFFC 00000							
			5B	00000000G	00	9E	00002				
			5A	00000000G	00	9E	00009				
			59	00000000G	00	9E	00010				
			58	00000000G	00	9E	00017				
			57	00000000G	00	9E	0001E				
			56	0000	CF	9E	00025				
			5E	FF4C	CE	9E	0002A				
54	AE	00C0	C6	004C	8F	28	0002F				
					55	D4	00038				
				08	AE	D4	0003A				
		74	AE	020E0000	8F	D0	0003D				
				78	AE	D4	00045				
				74	AE	9F	00048				
				011C	C6	9F	0004B				
		00000000G	00		02	FB	0004F				
			0D		50	E8	00056				
			55		01	D0	00059				
		74	AE		0B	B0	0005C				
		78	AE	0124	C6	9E	00060				
			50	0C	AE	9E	00066	1\$:			
			80	00040004	8F	D0	0006A				
			80		6E	9E	00071				
					80	D4	00074				
			80	00020004	8F	D0	00076				
			80	04	AE	9E	0007D				
					80	D4	00081				
			80	00200014	8F	D0	00083				
			80	EC	AD	9E	0008A				
			80	74	AE	9E	0008E				
					80	D4	00092				
					7E	7C	00094				
					7E	D4	00096				
				50	AE	9F	00098				
				1C	AE	9F	0009B				
				C0	AD	9F	0009E				
					7E	7C	000A1				
		00000000G	00		08	FB	000A3				
			54		50	D0	000AA				
			76		54	E9	000AD				
			54	44	AE	3C	000B0				

.EXTRN	TERMS_VT173, TERMS_VT52	
.EXTRN	TERMS_VT200 SERIES	
.EXTRN	TERMS_PRO SERIES	
.EXTRN	TERMS_FT1, TERMS_FT8	
.EXTRN	TERMS_UNKNOWN, SETS_TERMSET	
.EXTRN	SETS_WRITEERR, SETS_UNKTERM	
.EXTRN	SETS_NOPERM, CLIS_IDEVTYPE	
.EXTRN	CLIS_ABSENT, CLIS_NEGATED	
.EXTRN	CLIS_PRESENT, SYSSGETDVIW	
.EXTRN	SYSS\$ASSIGN, SYSS\$GETJPI	
.EXTRN	SYSS\$CMKRN, SYSS\$QIOW	
.EXTRN	SYSS\$SETIMR, SYSS\$CANTIM	
.PSECT	\$CODE\$,NOWRT,2	
.ENTRY	SET\$TERMINAL, Save R2,R3,R4,R5,R6,R7,R8,R9,-;	0442
	R10,R11	
MOVAB	SYSS\$CANTIM, R11	
MOVAB	SYSS\$SETIMR, R10	
MOVAB	LIB\$SIGNAL, R9	
MOVAB	CLIS\$PRESENT, R8	
MOVAB	SYSS\$QIOW, R7	
MOVAB	SD_PROTOCOL, R6	
MOVAB	-180(SP), SP	
MOVAB	#76, P.ABC, DATA_BUFFER	0465
CLRL	DEFAULT_DEVICE	
CLRL	REFCOUNT	
MOVL	#34471936, DEV_DESC	0484
CLRL	DEV_DESC+4	
PUSHAB	DEV_DESC	0486
PUSHAB	P.ABE	
CALLS	#2, CLIS\$GET_VALUE	
BLBS	R0, 1\$	
MOVL	#1, DEFAULT_DEVICE	0490
MOVW	#11, DEV_DESC	0491
MOVAB	P.ABG, DEV_DESC+4	0492
MOVAB	DVI_LIST, \$\$ITMBLKPTR	0510
MOVL	#262148, (\$\$ITMBLKPTR)+	
MOVAB	CLASS, (\$\$ITMBLKPTR)+	
CLRL	(\$\$ITMBLKPTR)+	
MOVL	#131076, (\$\$ITMBLKPTR)+	
MOVAB	DEV_CHAR, (\$\$ITMBLKPTR)+	
CLRL	(\$\$ITMBLKPTR)+	
MOVL	#2097172, (\$\$ITMBLKPTR)+	
MOVAB	DEV_BUFFER, (\$\$ITMBLKPTR)+	
MOVAB	DEV_DESC, (\$\$ITMBLKPTR)+	
CLRL	(\$\$ITMBLKPTR)+	
CLRQ	-(SP)	0513
CLRL	-(SP)	
PUSHAB	IOSB	
PUSHAB	DVI_LIST	
PUSHAB	DEV_DESC	
CLRQ	-(SP)	
CALLS	#8, SYSS\$GETDVIW	
MOVL	R0, STATUS	
BLBC	STATUS, 4\$	0514
MOVZWL	IOSB, STATUS	0515

	6F		54	E9	000B4	BLBC	STATUS, 4\$	0516
		74	AE	B7	000B7	DECW	DEV_DESC	0525
78	AE	ED	AD	9E	000BA	MOVAB	DEV_BUFFER+1, DEV_DESC+4	0526
00000042	8F		6E	D1	000BF	CMPL	CLASS, #66	0527
			08	13	000C6	BEQL	2\$	
		00000000G	8F	DD	000C8	PUSHL	#CLIS_IVDEVTYPE	0530
			60	11	000CE	BRB	5\$	
			7E	7C	000D0	CLRQ	-(SP)	0542
		D8	AD	9F	000D2	PUSHAB	CHAN	
		CO	AD	9F	000D5	PUSHAB	DEV_DESC	
00000000G	00		04	FB	000D8	CALLS	#4, -SYSSASSIGN	
	54		50	D0	000DF	MOVL	R0, STATUS	
	5D		54	E9	000E2	BLBC	STATUS, 7\$	
			56	DD	000E5	PUSHL	R6	0559
	68		01	FB	000E7	CALLS	#1, CLISPRESNT	
	03		50	E8	000EA	BLBS	R0, 3\$	
			00BB	31	000ED	BRW	11\$	
	50	14	AE	9E	000F0	MOVAB	GETJPI_LIST, \$\$ITMBLKPTR	0567
	80	02040004	8F	D0	000F4	MOVL	#33816580, (\$\$ITMBLKPTR)+	
	80	24	AE	9E	000FB	MOVAB	PROCESS_PRIVS, (\$\$ITMBLKPTR)+	
			80	7C	000FF	CLRQ	(\$\$ITMBLKPTR)+	
2C	AE	020E0000	8F	D0	00101	MOVL	#34471936, DESC	0568
		30	AE	D4	00109	CLRL	DESC+4	
71	AE	40	8F	8A	0010C	BICB2	#64, FLAGS+1	0572
			7E	7C	00111	CLRQ	-(SP)	0574
			7E	D4	00113	CLRL	-(SP)	
		20	AE	9F	00115	PUSHAB	GETJPI_LIST	
			7E	7C	00118	CLRQ	-(SP)	
			7E	D4	0011A	CLRL	-(SP)	
00000000G	00		07	FB	0011C	CALLS	#7, SYSSGETJPI	
	54		50	D0	00123	MOVL	R0, STATUS	
05	19		54	E9	00126	BLBC	STATUS, 7\$	0575
26	AE		02	E0	00129	BBS	#2, PROCESS_PRIVS+2, 6\$	0582
			24	DD	0012E	PUSHL	#36	0585
			028E	31	00130	BRW	35\$	
		2C	AE	9F	00133	PUSHAB	DESC	0589
			56	DD	00136	PUSHL	R6	
00000000G	00		02	FB	00138	CALLS	#2, CLISGET_VALUE	
	03		50	E8	0013F	BLBS	R0, 8\$	
			027A	31	00142	BRW	34\$	
	50	00000000G	00	D0	00145	MOVL	PROTOCOL\$ NONE+4, R0	0594
60	30	BE	2C	AE	29	CMPC3	DESC, @DESC+4, (R0)	0593
			25	12	00152	BNEQ	9\$	
	0C	AE	01	D0	00154	MOVL	#1, ARGLIST	0597
	10	AE	AD	3C	00158	MOVZWL	CHAN, ARGLIST+4	0598
		D8	AE	9F	0015D	PUSHAB	ARGLIST	0600
		0C	00	9F	00160	PUSHAB	SWITCH TO TERMINAL	
00000000G	00	00000000G	02	FB	00166	CALLS	#2, SYSSCMKRN	
	54		50	D0	0016D	MOVL	R0, STATUS	
	CF		54	E9	00170	BLBC	STATUS, 7\$	
	05	AE	20	8A	00173	BICB2	#32, DEV_CHAR+1	0612
			32	11	00177	BRB	11\$	0591
	50	00000000G	00	D0	00179	MOVL	PROTOCOL\$ DDCMP+4, R0	0622
60	30	BE	2C	AE	29	CMPC3	DESC, @DESC+4, (R0)	0621
			07	12	00186	BNEQ	10\$	
	71	AE	8F	88	00188	BISB2	#64, FLAGS+1	0625
		40	1C	11	0018D	BRB	11\$	0615

			30	56	DD	0018F	10\$:	PUSHL	R6		0629
				AE	9F	00191		PUSHAB	DESC		
			0077132A	02	DD	00194		PUSHL	#2		
			C0	8F	DD	00196		PUSHL	#7803690		
				AD	9F	0019C		PUSHAB	DEV_DESC		
				01	DD	0019F		PUSHL	#1		
			00000000*	8F	DD	001A1		PUSHL	#<SET\$ WRITEERR-8>		0630
		69		07	FB	001A7		CALLS	#7, LIB\$SIGNAL		
					04	001AA		RET			0628
08	05	AE		05	E1	001AB	11\$:	BBC	#5, DEV_CHAR+1, 12\$		0648
		7E	228C	8F	3C	001B0		MOVZWL	#8844, =(SP)		0651
				0209	31	001B5		BRW	35\$		
			013C	C6	9F	001B8	12\$:	PUSHAB	P.ABH		0659
		68		01	FB	001BC		CALLS	#1, CLISPRESENT		
70	AE	01		50	FO	001BF		INSV	R0, #1, #1, FLAGS		
			0148	C6	9F	001C5		PUSHAB	P.ABJ		0660
		68		01	FB	001C9		CALLS	#1, CLISPRESENT		
70	AE	01		50	FO	001CC		INSV	R0, #0, #1, FLAGS		
		00		AE	9E	001D2		MOVAB	DVI_LIST2, \$\$ITMBLKPTR		0672
		50	34	8F	DO	001D6		MOVL	#1966084, (\$\$ITMBLKPTR)+		
		80	001E0004	AE	9E	001DD		MOVAB	REFCOUNT, (\$\$ITMBLKPTR)+		
		80	08	7E	7C	001E1		CLRQ	(\$\$ITMBLKPTR)+		
				7E	7C	001E3		CLRQ	-(SP)		0673
				7E	D4	001E5		CLRL	-(SP)		
			50	AE	9F	001E7		PUSHAB	IOSB		
			44	AE	9F	001EA		PUSHAB	DVI_LIST2		
			C0	AD	9F	001ED		PUSHAB	DEV_DESC		
				7E	7C	001F0		CLRQ	-(SP)		
		00000000G	00	08	FB	001F2		CALLS	#8, SYSSGETDVIW		
			54	50	DO	001F9		MOVL	R0, STATUS		
			52	54	E9	001FC		BLBC	STATUS, 16\$		0674
			54	44	AE	3C	001FF	MOVZWL	IOSB, STATUS		0675
			48	54	E9	00203		BLBC	STATUS, 16\$		0676
			01	08	AE	D1	00206	CMPL	REFCOUNT, #1		0685
				19	12	0020A		BNEQ	13\$		
				55	E8	0020C		BLBS	DEFAULT DEVICE, 13\$		0687
11	70	AE		01	E0	0020F		BBS	#1, FLAGS, 13\$		0689
				56	DD	00214		PUSHL	R6		0691
		68		01	FB	00216		CALLS	#1, CLISPRESENT		
		09		50	E8	00219		BLBS	R0, 13\$		
			00000000*	8F	DD	0021C		PUSHL	#<SET\$ NOPERM-8>		0693
		69		01	FB	00222		CALLS	#1, LIB\$SIGNAL		
				7E	7C	00225	13\$:	CLRQ	-(SP)		0704
				7E	7C	00227		CLRQ	-(SP)		
				0C	DD	00229		PUSHL	#12		
			C8	AD	9F	0022B		PUSHAB	INFO_BLOCK		
				7E	7C	0022E		CLRQ	-(SP)		
			64	AE	9F	00230		PUSHAB	IOSB		
04	BC	AD		01	E1	00233		BBC	#1, FLAGS, 14\$		
				1B	DD	00238		PUSHL	#27		
				02	11	0023A		BRB	15\$		
				27	DD	0023C	14\$:	PUSHL	#39		
		7E	D8	AD	3C	0023E	15\$:	MOVZWL	CHAN, -(SP)		
				7E	D4	00242		CLRL	-(SP)		
		67		0C	FB	00244		CALLS	#12, SYSSQIOW		
		54		50	DO	00247		MOVL	R0, STATUS		
		04		54	E9	0024A		BLBC	STATUS, 16\$		0705

50	D3	2B	D3	54	44	AE	3C	0024D	MOVZWL	IOSB, STATUS	0706	
				03		54	E8	00251	16\$:	BLBS	STATUS, 17\$	0707
					0168	31	00254			BRW	34\$	
					68	AE	7C	00257	17\$:	CLRQ	PARITY	0717
				0000V	54	AE	9F	0025A		PUSHAB	DATA BUFFER	0724
				08		01	FB	0025D		CALLS	#1, GET_TERM_TYPE	
						50	E9	00262		BLBC	RO, 18\$	
				0000V	54	AE	9F	00265		PUSHAB	DATA BUFFER	0731
				01		01	FB	00268		CALLS	#1, GET_VALUES	
						50	E8	0026D	18\$:	BLBS	RO, 19\$	
							04	00270		RET		
				30	71	AE	E9	00271	19\$:	BLBC	FLAGS+1, 21\$	0733
				AD		05	E1	00275		BBC	#5, INFO_BLOCK+11, 21\$	0734
				01		03	EF	0027A		EXTZV	#3, #1, INFO_BLOCK+11, RO	0739
				50		50	D2	00280		MCOML	RO, RO	
				0050	7E	AE	B1	00283		CMPW	INFO_BLOCK+2, #80	0735
				8F		0E	1B	00289		BLEQU	20\$	
					0150	C6	9E	0028B		MOVAB	P.ABL, SET_LENGTH	0738
				52		50	E9	00290		BLBC	RO, 21\$	0739
				12		0E	90	00293		MOVB	#14, INFO_BLOCK+7	0740
				CF		0C	11	00297		BRB	21\$	0735
					0155	C6	9E	00299	20\$:	MOVAB	P.ABM, SET_LENGTH	0744
				52		50	E9	0029E		BLBC	RO, 21\$	0745
				04		18	90	002A1		MOVB	#24, INFO_BLOCK+7	0746
				CF		7E	D4	002A5	21\$:	CLRL	-(SP)	0762
						6C	AE	DD	002A7	PUSHL	PARITY	
						74	AE	DD	002AA	PUSHL	FILL	
						70	AE	DD	002AD	PUSHL	SPEED	
							0C	DD	002B0	PUSHL	#12	
						C8	AD	9F	002B2	PUSHAB	INFO_BLOCK	
						7E	7C	002B5		CLRQ	-(SP)	
				04	64	AE	9F	002B7		PUSHAB	IOSB	
				BC		01	E1	002BA		BBC	#1, FLAGS, 22\$	
				AD		1A	DD	002BF		PUSHL	#26	
						02	11	002C1		BRB	23\$	
						23	DD	002C3	22\$:	PUSHL	#35	
				7E	D8	AD	3C	002C5	23\$:	MOVZWL	CHAN, -(SP)	
						7E	D4	002C9		CLRL	-(SP)	
				67		0C	FB	002CB		CALLS	#12, SYSSQIOW	
				54		50	D0	002CE		MOVL	RO, STATUS	
				04		54	E9	002D1		BLBC	STATUS, 24\$	0763
				54	44	AE	3C	002D4		MOVZWL	IOSB, STATUS	0764
				8F		54	D1	002D8	24\$:	CMP	STATUS, #1689	0765
						05	12	002DF		BNEQ	25\$	
						54	DD	002E1		PUSHL	STATUS	0766
				69		01	FB	002E3		CALLS	#1, LIBSSIGNAL	
				6F		54	E9	002E6	25\$:	BLBC	STATUS, 27\$	0767
				AE		06	E1	002E9		BBC	#6, FLAGS+1, 26\$	0778
				2C		01	D0	002EE		MOVL	#1, ARGLIST	0782
				30		AD	3C	002F2		MOVZWL	CHAN, ARGLIST+4	0783
						AE	9F	002F7		PUSHAB	ARGLIST	0785
						00	9F	002FA		PUSHAB	SWITCH TO DDCMP	
				00000000G		02	FB	00300		CALLS	#2, SYSSCMKRN	
				54		50	D0	00307		MOVL	RO, STATUS	
				4B		54	E9	0030A		BLBC	STATUS, 27\$	
							04	0030D		RET		0789
				49	71	AE	E9	0030E	26\$:	BLBC	FLAGS+1, 28\$	0799

49	D3	AD		05	E1	00312	BBC	#5, INFO_BLOCK+11, 29\$	0800
		7E	D8	AD	3C	00317	MOVZWL	CHAN, -(SP)	0806
			0000V	CF	9F	0031B	PUSHAB	WRITE TIMEOUT	
			010C	C6	9F	0031F	PUSHAB	TIMEOUT	
				7E	D4	00323	CLRL	-(SP)	
		6A		04	FB	00325	CALLS	#4, SYSS\$SETIMR	
				7E	7C	00328	CLRQ	-(SP)	0812
				7E	7C	0032A	CLRQ	-(SP)	
				05	DD	0032C	PUSHL	#5	
				52	DD	0032E	PUSHL	SET LENGTH	
				7E	7C	00330	CLRQ	-(SP)	
			64	AE	9F	00332	PUSHAB	IOSB	
				30	DD	00335	PUSHL	#48	
		7E	D8	AD	3C	00337	MOVZWL	CHAN, -(SP)	
				7E	D4	0033B	CLRL	-(SP)	
		67		0C	FB	0033D	CALLS	#12, SYSS\$QIOW	
		54		50	D0	00340	MOVL	RO, STATUS	
				7E	D4	00343	CLRL	-(SP)	0814
		7E	D8	AD	3C	00345	MOVZWL	CHAN, -(SP)	
		6B		02	FB	00349	CALLS	#2, SYSS\$CANTIM	
		70		54	E9	0034C	BLBC	STATUS, 34\$	0815
			44	AE	B5	0034F	TSTW	IOSB	0816
				04	13	00352	BEQL	27\$	
		54	44	AE	3C	00354	MOVZWL	IOSB, STATUS	
		64		54	E9	00358	BLBC	STATUS, 34\$	0817
0A	D3	AD		05	E0	0035B	BBS	#5, INFO_BLOCK+11, 30\$	0825
	00000000*	00	7D	AE	91	00360	CMPB	INFO_BLOCK+1, <TERMS_TABLE+<<TERMS_VT52*28>-	0826
								+1>>	
		7E	D8	66	12	00368	BNEQ	36\$	
			0000V	AD	3C	0036A	MOVZWL	CHAN, -(SP)	0833
			010C	CF	9F	0036E	PUSHAB	WRITE TIMEOUT	
				C6	9F	00372	PUSHAB	TIMEOUT	
				7E	D4	00376	CLRL	-(SP)	
		6A		04	FB	00378	CALLS	#4, SYSS\$SETIMR	
				7E	7C	0037B	CLRQ	-(SP)	0840
				7E	7C	0037D	CLRQ	-(SP)	
				02	DD	0037F	PUSHL	#2	
			D2	AD	95	00381	TSTB	INFO_BLOCK+10	
				07	18	00384	BGEQ	31\$	
		50	015A	C6	9E	00386	MOVAB	P.ABN, RO	
				05	11	0038B	BRB	32\$	
		50	015C	C6	9E	0038D	MOVAB	P.ABO, RO	
				50	DD	00392	PUSHL	RO	
				7E	7C	00394	CLRQ	-(SP)	
			64	AE	9F	00396	PUSHAB	IOSB	
				30	DD	00399	PUSHL	#48	
		7E	D8	AD	3C	0039B	MOVZWL	CHAN, -(SP)	
				7E	D4	0039F	CLRL	-(SP)	
		67		0C	FB	003A1	CALLS	#12, SYSS\$QIOW	
		54		50	D0	003A4	MOVL	RO, STATUS	
				7E	D4	003A7	CLRL	-(SP)	0842
		7E	D8	AD	3C	003A9	MOVZWL	CHAN, -(SP)	
		6B		02	FB	003AD	CALLS	#2, SYSS\$CANTIM	
		0C		54	E9	003B0	BLBC	STATUS, 34\$	0843
			44	AE	B5	003B3	TSTW	IOSB	0844
				04	13	003B6	BEQL	33\$	
		54	44	AE	3C	003B8	MOVZWL	IOSB, STATUS	

11		54	E8	003BC	33\$:	BLBS	STATUS, 36\$:	0845	
		54	DD	003BF	34\$:	PUSHL	STATUS	:	0849	
	78	AE	9F	003C1	35\$:	PUSHAB	DEV_DESC	:	0848	
		01	DD	003C4		PUSHL	#1	:		
	00000000*	8F	DD	003C6		PUSHL	#<SET\$ WRITEERR8-8>	:		
69		04	FB	003CC		CALLS	#4, LIB\$SIGNAL	:		
			04	003CF		RET		:	0847	
37	71	AE	05	E1	003D0	36\$:	BBC	#5, FLAGS+1, 37\$:	0853
		7E	AD	3C	003D5		MOVZWL	CHAN, -(SP)	:	0858
			CF	9F	003D9		PUSHAB	WRITE_TIMEOUT	:	
			C6	9F	003DD		PUSHAB	TIMEOUT	:	
			7E	D4	003E1		CLRL	-(SP)	:	
	6A		04	FB	003E3		CALLS	#4, SYSS\$SETIMR	:	
			7E	7C	003E6		CLRQ	-(SP)	:	0864
			7E	7C	003E8		CLRQ	-(SP)	:	
			09	DD	003EA		PUSHL	#9	:	
		015E	C6	9F	003EC		PUSHAB	P.ABP	:	
			7E	7C	003F0		CLRQ	-(SP)	:	
		64	AE	9F	003F2		PUSHAB	IOSB	:	
			30	DD	003F5		PUSHL	#48	:	
	7E	D8	AD	3C	003F7		MOVZWL	CHAN, -(SP)	:	
			7E	D4	003FB		CLRL	-(SP)	:	
67			0C	FB	003FD		CALLS	#12, SYSS\$QIOW	:	
54			50	D0	00400		MOVL	R0, STATUS	:	
			7E	D4	00403		CLRL	-(SP)	:	0866
	7E	D8	AD	3C	00405		MOVZWL	CHAN, -(SP)	:	
6B			02	FB	00409		CALLS	#2, SYSS\$CANTIM	:	
08		70	AE	E9	0040C	37\$:	BLBC	FLAGS, 38\$:	0868
		54	AE	9F	00410		PUSHAB	DATA_BUFFER	:	0869
0000V	CF		01	FB	00413		CALLS	#1, LOG_RESULTS	:	
			04	00418	38\$:	RET		:	0872	

; Routine Size: 1049 bytes, Routine Base: \$CODE\$ + 0000

; 778 0873 1


```
: 780      0874 1 ROUTINE write_timeout(chan): NOVALUE =
: 781      0875 1
: 782      0876 1 ---
: 783      0877 1
: 784      0878 1 The timeout has elapsed while trying to write to the primary
: 785      0879 1 output stream. We assume that the user pressed control/s to
: 786      0880 1 inhibit completion of the write. Cancel the I/O to force
: 787      0881 1 completion of the $QIO.
: 788      0882 1
: 789      0883 1 Inputs:
: 790      0884 1
: 791      0885 1 CHAN = The channel to cancel io on. Since this is called
: 792      0886 1 from a timer ast the channel must be the request id.
: 793      0887 1
: 794      0888 1 Outputs:
: 795      0889 1
: 796      0890 1 None
: 797      0891 1 ---
: 798      0892 2 BEGIN
: 799      0893 2 $CANCEL(CHAN = .chan); ! Cancel the I/O to the terminal
: 800      0894 1 END;
```

.EXTRN SYSS\$CANCEL

0000 00000 WRITE_TIMEOUT:

```
00000000G 00      04 AC DD 00002      .WORD Save nothing
                  01 FB 00005      PUSHL CHAN
                  04 0000C      CALLS #1, SYSS$CANCEL
                        RET
```

: 0874
: 0893
: 0894

; Routine Size: 13 bytes, Routine Base: \$CODE\$ + 0419

```

: 802      0895 1 ROUTINE get_term_type (data_buffer) =
: 803      0896 2 BEGIN
: 804      0897 3
: 805      0898 4 ++
: 806      0899 5 Functional description
: 807      0900 6
: 808      0901 7     This routine determines if a specific device type was specified,
: 809      0902 8     and, if it was, sets the associated fields in the device
: 810      0903 9     characteristics buffer.
: 811      0904 10
: 812      0905 11
: 813      0906 12 Inputs
: 814      0907 13     DATA_BUFFER - contains all the meaningful data
: 815      0908 14
: 816      0909 15 Outputs
: 817      0910 16     INFO_BLOCK - will be changed according to what device type was
: 818      0911 17     INDEX      - will be set to a number corresponding to the term type
: 819      0912 18
: 820      0913 19
: 821      0914 20 -----
: 822      0915 21
: 823      0916 22 MAP
: 824      0917 23     data_buffer : REF VECTOR;
: 825      0918 24
: 826      0919 25
: 827      0920 26 Bind the data buffer to names we understand.
: 828      0921 27
: 829      0922 28 bind_data;
: 830      0923 29
: 831      0924 30 BUILTIN
: 832      0925 31     CALLG,
: 833      0926 32     AP;
: 834      0927 33
: 835      0928 34 LOCAL
: 836      0929 35     status;
: 837      0930 36
: 838      0931 37 index = -1;
: 839      0932 38
: 840      0933 39
: 841      0934 40 See if /INQUIRE was specified.
: 842      0935 41
: 843      0936 42 IF cli$present(%ASCII 'INQUIRE')
: 844      0937 43 THEN
: 845      0938 44     BEGIN
: 846      0939 45     IF NOT inquire_type(.data_buffer)
: 847      0940 46     THEN RETURN 0;
: 848      0941 47     END;
: 849      0942 48
: 850      0943 49
: 851      0944 50 See if a specific devtype was mentioned. Before the /DEVICE_TYPE
: 852      0945 51 qualifier was implemented, it was possible to specify a devtype just
: 853      0946 52 by calling it out, e.g. /VT100. So, to stay compatible with earlier
: 854      0947 53 versions, see if any of those were specified.
: 855      0948 54
: 856      0949 55 IF .index EQL -1
: 857      0950 56 THEN INCR i FROM 0 TO 14 DO
: 858      0951 57     (IF cli$present(.term$_name[i])
```



```
859 0952 2 THEN (index = .i; EXITLOOP));
860 0953
861 0954
862 0955 The newer method gets a value for DEVICE_TYPE. So, if nothing found
863 0956 yet, try /DEVICE_TYPE.
864 0957
865 0958 IF .index EQL -1
866 0959 THEN
867 0960 BEGIN
868 0961 $init_dyndesc(name_desc);
869 0962 IF cli$get_value(SD_DEVICE_TYPE, name_desc)
870 0963 THEN
871 0964 BEGIN
872 0965 DECR i FROM term$num - 1 TO 0 DO
873 0966 BEGIN
874 0967 BIND name = .term$name[i] : $BLOCK;
875 0968 IF CH$EQL(.name_desc[dsc$w_length], .name_desc[dsc$a_pointer],
876 0969 .name_desc[dsc$w_length], .name_desc[dsc$a_pointer])
877 0970 THEN (index = .i; EXITLOOP);
878 0971 END;
879 0972 IF .index EQL -1
880 0973 THEN
881 0974 BEGIN
882 0975
883 0976 Didn't find a known type. See if this terminal is
884 0977 defined in TERMTABLE.EXE, the terminal definition file.
885 0978
886 0979 status = CALLG (.AP, get_term_def);
887 0980 IF NOT .status
888 0981 THEN
889 0982 RETURN 0 ! errors already signalled
890 0983 ELSE
891 0984 RETURN 1; ! get_term_def stored type, width, etc.
892 0985 END;
893 0986 END;
894 0987
895 0988
896 0989
897 0990 If a device type was specified, set it. But first, a word about the
898 0991 way that things get set or don't get set.
899 0992
900 0993 If the device type was specified as FT1 thru FT8, simply set the type
901 0994 field.
902 0995
903 0996 If /UNKNOWN, then set the type field and clear some bits in the second
904 0997 characteristics longword.
905 0998
906 0999 IF .index NEQ -1
907 1000 THEN
908 1001 BEGIN
909 1002 LOCAL
910 1003 mask;
911 1004 info_block[term$b_type] = .term$table[index, term$b_type];
912 1005 IF .index GEQ term$ft1
913 1006 AND .index LEQ term$ft8
914 1007 THEN RETURN 1;
915 1008
```



```

: 916      1009      mask = .term$_table[.index, term$l_set2] OR
: 917      1010      .term$-table[.index, term$l_clr2];
: 918      1011      info_block[term$l_set2] = .info_block[term$l_set2]
: 919      1012      AND NOT .mask
: 920      1013      OR .term$-table[.index, term$l_set2];
: 921      1014      IF .index EQL term$_unknown
: 922      1015      THEN RETURN 1;
: 923      1016
: 924      1017      info_block[term$_width] = .term$-table[.index, term$_width];
: 925      1018      info_block[term$_page] = .term$-table[.index, term$_page];
: 926      1019      mask = .term$-table[.index, term$l_set1] OR
: 927      1020      .term$-table[.index, term$l_clr1];
: 928      1021      info_block[term$l_set1] = .info_block[term$l_set1]
: 929      1022      AND NOT .mask
: 930      1023      OR .term$-table[.index, term$l_set1];
: 931      1024      END;
: 932      1025
: 933      1026      RETURN 1;
: 934      1027      END;

```

.PSECT SPLITS, NOWRT, NOEXE, 2

```

00 45 52 49 55 51 4E 49 0017F .BLKB 1
010E0007 00180 P.ABR: .ASCII \INQUIRE\<0>
00000000 00188 P.ABQ: .LONG 17694727
00000000 0018C .ADDRESS P.ABR

```

.PSECT \$CODE\$, NOWRT, 2

```

07FC 0000 GET_TERM_TYPE:
5A 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10
59 00000000G 00 9E 00009 MOVAB TERMS_NAME, R10
58 00000000G 00 9E 00010 MOVAB CLISPRESNT, R9
52 04 AC D0 00017 MOVAB TERMS_TABLE+1, R8
55 28 A2 9E 0001B MOVAB DATA_BUFFER, R2
57 34 A2 9E 0001F MOVAB 40(R2), R5
54 44 A2 9E 00023 MOVAB 52(R2), R7
67 01 CE 00027 MOVAB 68(R2), R4
0000' CF 9F 0002A MNEGL #1, (R7)
69 01 FB 0002E PUSHAB P.ABQ
0A 50 E9 00031 CALLS #1, CLISPRESNT
52 DD 00034 BLBC R0, 1$
0000V CF 01 FB 00036 PUSHL R2
6F 50 E9 0003B CALLS #1, INQUIRE_TYPE
FFFFFFF 8F 67 D1 0003E 1$: BLBC R0, 8$
14 12 00045 CMPL (R7), #-1
52 D4 00047 BNEQ 4$
6A42 DD 00049 2$: CLRL I
69 01 FB 0004C PUSHL TERMS_NAME[I]
05 50 E9 0004F CALLS #1, CLISPRESNT
67 52 D0 00052 BLBC R0, 3$
04 11 00055 MOVBL I, (R7)
BRB 4$

```


EE	FFFFFFF	52	0E	F3	00057	3\$:	AOBLEQ	#14, I, 2\$	0951	
		8F	67	D1	00058	4\$:	CMPL	(R7), #-1	0958	
			4B	12	00062		BNEQ	9\$		
		64	8F	D0	00064		MOVL	#34471936, (R4)	0961	
			A4	D4	0006B		CLRL	4(R4)		
			54	DD	0006E		PUSHL	R4	0962	
			CF	9F	00070		PUSHAB	SD_DEVICE_TYPE		
	00000000G	00	02	FB	00074		CALLS	#2, CLISGET_VALUE		
		31	50	E9	0007B		BLBC	R0, 9\$		
56	00000000G	8F	01	C1	0007E		ADDL3	#1, #TERMS_NUM-1, I	0965	
			11	11	00086		BRB	6\$		
		50	6A	46	D0	00088	5\$:	MOVL	TERMS_NAME[I], R0	0967
04	B0	04	64	29	0008C		CMPC3	(R4), @4(R4), @4(R0)	0968	
			05	12	00092		BNEQ	6\$		
		67	56	D0	00094		MOVL	I, (R7)	0970	
			03	11	00097		BRB	7\$		
		EC	56	F4	00099	6\$:	SOBGEQ	I, 5\$	0965	
	FFFFFFF	8F	67	D1	0009C	7\$:	CMPL	(R7), #-1	0972	
			0A	12	000A3		BNEQ	9\$		
			6C	FA	000A5		CALLG	(AP), GET_TERM_DEF	0979	
	0000V	CF	50	E8	000AA		BLBS	STATUS, 1T\$	0980	
		6F	71	11	000AD	8\$:	BRB	12\$	0984	
	FFFFFFF	8F	67	D1	000AF	9\$:	CMPL	(R7), #-1	0999	
			64	13	000B6		BEQL	11\$		
50		67	1C	C5	000B8		MULL3	#28, (R7), R0	1004	
	01	A5	68	40	90	000BC	MOVB	TERMS_TABLE+1[R0], 1(R5)		
	00000000G	8F	67	D1	000C1		CMPL	(R7), #TERMS_FT1	1005	
			09	19	000C8		BLSS	10\$		
	00000000G	8F	67	D1	000CA		CMPL	(R7), #TERMS_FT8	1006	
			49	15	000D1		BLEQ	11\$		
			07	A8	40	9F	000D3	10\$:		
		52	9E	D0	000D7		PUSHAB	TERMS_TABLE+8[R0]	1009	
			0F	A8	40	9F	000DA			
			9E	C9	000DE		MOVL	@(SP)+, R2	1010	
53		52	53	CB	000E2		PUSHAB	TERMS_TABLE+16[R0]		
51	08	A5	52	C9	000E7		BISL3	@(SP)+, R2, MASK	1012	
08	A5	51	52	C9	000E7		BICL3	MASK, 8(R5), R1	1013	
	00000000G	8F	67	D1	000EC		BISL3	R2, R1, 8(R5)	1014	
			27	13	000F3		CMPL	(R7), #TERMS_UNKNOWN		
			01	A8	40	9F	000F5			
		02	9E	B0	000F9		BEQL	11\$	1017	
		07	06	A8	40	90	000FD			
			03	A8	40	9F	00103		1018	
			9E	D0	00107		PUSHAB	TERMS_TABLE+4[R0]	1019	
		51	0B	A8	40	9F	0010A			
			9E	C9	0010E		MOVL	@(SP)+, R1	1020	
53		51	53	CB	00112		PUSHAB	TERMS_TABLE+12[R0]		
50	04	A5	51	C9	00117		BISL3	@(SP)+, R1, MASK	1022	
			01	D0	0011C	11\$:	BICL3	MASK, 4(R5), R0	1023	
			04	04	0011F		BISL3	R1, R0, 4(R5)	1026	
			50	D4	00120	12\$:	MOVL	#1, R0		
			04	04	00122		RET		1027	
							CLRL	R0		
							RET			

; Routine Size: 291 bytes, Routine Base: \$CODE\$ + 0426


```

936 1028 1 ROUTINE get_values (data_buffer) =
937 1029 2 BEGIN
938 1030 3
939 1031 4 ++
940 1032 5 Functional description
941 1033 6
942 1034 7 This routine interrogates the CLI to obtain values for various
943 1035 8 terminal characteristics.
944 1036 9
945 1037 10 Inputs
946 1038 11 CHAN - channel which is assigned to the terminal
947 1039 12
948 1040 13 Outputs
949 1041 14 INFO_BLOCK - changed to reflect qualifiers
950 1042 15 TT1 - tells what was set/cleared in 1st characteristic longword
951 1043 16 TT2 - tells what was set/cleared in 2nd characteristic longword
952 1044 17 SPEED - contains the new speed values
953 1045 18 PARITY - contains new parity code
954 1046 19 FILL - contains new fill characteristics
955 1047 20 FLAGS - bits set to show what was changed
956 1048 21
957 1049 22 ----
958 1050 23
959 1051 24 MAP
960 1052 25 data_buffer : REF VECTOR;
961 1053 26
962 1054 27 Bind the data buffer to names we understand.
963 1055 28
964 1056 29 bind_data;
965 1057 30
966 1058 31 LOCAL
967 1059 32
968 1060 33 crfill,
969 1061 34 decrt,
970 1062 35 lffill,
971 1063 36 desc : $BBLOCK[dsc$c_s_bln],
972 1064 37 status,
973 1065 38 status2 ;
974 1066 39
975 1067 40 $init_dyndesc(desc); ! Make a dynamic descriptor
976 1068 41
977 1069 42 Parity.
978 1070 43
979 1071 44 status = cli$present (SD_PARITY);
980 1072 45 IF .status ! If present,
981 1073 46 THEN
982 1074 47 BEGIN ! say that we want to change
983 1075 48 parity = .parity OR tt$m_parity OR tt$m_altrpar; ! parity, assume even.
984 1076 49 IF cli$get_value(SD_PARITY, desc) ! If a parity value given
985 1077 50 THEN
986 1078 51 BEGIN
987 1079 52 IF CH$EQL(.desc[dsc$w_length], .desc[dsc$a_pointer],
988 1080 53 .desc[dsc$w_length], .term$odd[T])
989 1081 54 THEN
990 1082 55 BEGIN
991 1083 56 parity = .parity OR tt$m_odd;
992 1084 57
```



```

993      1085 5      flags[set$V_odd] = 1;
994      1086 5      END
995      1087 4      ELSE IF CH$EQL(.desc[dsc$w_length], .desc[dsc$a_pointer],
996      1088 4      .desc[dsc$w_length], .term$_even[1])
997      1089 4      THEN flags[set$V_even] = 1
998      1090 4      ELSE IF CH$EQL(.desc[dsc$w_length], .desc[dsc$a_pointer],
999      1091 4      .desc[dsc$w_length], .term$_none[1])
1000      1092 4      THEN status = cli$_negated
1001      1093 4      ELSE
1002      1094 5      BEGIN
1003      1095 5      SIGNAL(set$_invquaval, 2, desc, SD_PARITY);
1004      1096 5      RETURN 0;
1005      1097 4      END;
1006      1098 3      END;
1007      1099 2      END;
1008      1100 2      IF .status EQL cli$_negated
1009      1101 2      THEN
1010      1102 3      BEGIN
1011      1103 3      parity = (.parity OR tt$m_altrpar) AND NOT (tt$m_parity OR tt$m_odd);
1012      1104 3      flags[set$V_nopar] = 1;
1013      1105 2      END;
1014      1106 2      !
1015      1107 2      frame size
1016      1108 2      !
1017      1109 2      IF cli$present (SD_FRAME)
1018      1110 2      THEN
1019      1111 3      BEGIN
1020      1112 3      LOCAL frame;
1021      1113 3      IF cli$get_value(SD_FRAME, desc)
1022      1114 3      THEN
1023      1115 4      BEGIN
1024      1116 4      IF NOT lib$cvdtb(.desc[dsc$w_length],
1025      1117 4      .desc[dsc$a_pointer],
1026      1118 4      frame)
1027      1119 4      THEN
1028      1120 5      BEGIN
1029      1121 5      SIGNAL(set$_invquaval, 2, desc, SD_FRAME);
1030      1122 5      RETURN 0;
1031      1123 4      END;
1032      1124 5      IF NOT (.frame EQL 0
1033      1125 6      OR (.frame GEQ 5
1034      1126 5      AND .frame LEQ 8))
1035      1127 4      THEN
1036      1128 5      BEGIN
1037      1129 5      SIGNAL(set$_invquaval, 2, desc, SD_FRAME);
1038      1130 5      RETURN 0;
1039      1131 4      END;
1040      1132 3      END;
1041      1133 3      parity = tt$m_altrframe or parity; ! say alter the frame size
1042      1134 3      parity <0,4> = .frame;
1043      1135 3      flags[set$V_frame] = 1;
1044      1136 2      end;
1045      1137 2      !
1046      1138 2      Page length
1047      1139 2      !
1048      1140 2      IF cli$present(SD_PAGE)
1049      1141 2      THEN
```

```
1050 1142 3 BEGIN
1051 1143 3 LOCAL page;
1052 1144 3 flags[set$v_page] = 1;
1053 1145 3 page = 0;
1054 1146 3 IF cli$get_value(SD_PAGE, desc)
1055 1147 3 THEN
1056 1148 4 BEGIN
1057 1149 4 IF NOT lib$cvdtb(.desc[dsc$w_length],
1058 1150 4 .desc[dsc$a_pointer],
1059 1151 4 page)
1060 1152 4 THEN
1061 1153 5 BEGIN
1062 1154 5 SIGNAL(set$_invquaval, 2, desc, SD_PAGE);
1063 1155 5 RETURN 0;
1064 1156 4 END;
1065 1157 4 IF .page LSS 0
1066 1158 4 OR .page GTR 255
1067 1159 4 THEN
1068 1160 5 BEGIN
1069 1161 5 SIGNAL(set$_invquaval, 2, desc, SD_PAGE);
1070 1162 5 RETURN 0;
1071 1163 4 END;
1072 1164 3 END;
1073 1165 3 info_block[term$b_page] = .page;
1074 1166 2 END;
1075 1167 2
1076 1168 2 Page width
1077 1169 2
1078 1170 2 IF cli$present(SD_WIDTH)
1079 1171 2 THEN
1080 1172 3 BEGIN
1081 1173 3 LOCAL width;
1082 1174 3 flags[set$v_width] = 1;
1083 1175 3 width = 0;
1084 1176 3 IF cli$get_value(SD_WIDTH, desc)
1085 1177 3 THEN
1086 1178 4 BEGIN
1087 1179 4 IF NOT lib$cvdtb(.desc[dsc$w_length],
1088 1180 4 .desc[dsc$a_pointer],
1089 1181 4 width)
1090 1182 4 THEN
1091 1183 5 BEGIN
1092 1184 5 SIGNAL(set$_invquaval, 2, desc, SD_WIDTH);
1093 1185 5 RETURN 0;
1094 1186 4 END;
1095 1187 4 IF .width LSS 0
1096 1188 4 OR .width GTR 511
1097 1189 4 THEN
1098 1190 5 BEGIN
1099 1191 5 SIGNAL(set$_invquaval, 2, desc, SD_WIDTH);
1100 1192 5 RETURN 0;
1101 1193 4 END;
1102 1194 3 END;
1103 1195 2 END;
1104 1196 2 info_block[term$w_width] = .width;
1105 1197 2 END;
1106 1198 2
```



```
1107 1199 2 |
1108 1200 2 | CRfill
1109 1201 2 |
1110 1202 2 | crfill = 0;
1111 1203 2 |
1112 1204 2 | IF cli$present(SD_CRFILL)
1113 1205 2 | THEN
1114 1206 2 | BEGIN
1115 1207 2 | flags[set$cr] = 1;
1116 1208 2 | IF cli$get_value(SD_CRFILL, desc)
1117 1209 2 | THEN
1118 1210 2 | BEGIN
1119 1211 2 | IF NOT lib$cvdtb(.desc[dsc$w_length],
1120 1212 2 | .desc[dsc$a_pointer],
1121 1213 2 | crfill)
1122 1214 2 | THEN
1123 1215 2 | BEGIN
1124 1216 2 | SIGNAL(set$_invquaval, 2, desc, SD_CRFILL);
1125 1217 2 | RETURN 0;
1126 1218 2 | END;
1127 1219 2 | END;
1128 1220 2 | IF .crfill LSS 0
1129 1221 2 | OR .crfill GTR 9
1130 1222 2 | THEN
1131 1223 2 | BEGIN
1132 1224 2 | SIGNAL(set$_invquaval, 2, desc, SD_CRFILL);
1133 1225 2 | RETURN 0;
1134 1226 2 | END;
1135 1227 2 | IF .crfill EQL 0
1136 1228 2 | THEN tt1_clr = .tt1_clr OR tt$m_crfill
1137 1229 2 | ELSE tt1_set = .tt1_set OR tt$m_crfill;
1138 1230 2 | fill<0,8> = .crfill
1139 1231 2 | END;
1140 1232 2 |
1141 1233 2 | LFfill
1142 1234 2 |
1143 1235 2 | lffill = 0;
1144 1236 2 |
1145 1237 2 | IF cli$present(SD_LFFILL)
1146 1238 2 | THEN
1147 1239 2 | BEGIN
1148 1240 2 | flags[set$lf] = 1;
1149 1241 2 | IF cli$get_value(SD_LFFILL, desc)
1150 1242 2 | THEN
1151 1243 2 | BEGIN
1152 1244 2 | IF NOT lib$cvdtb(.desc[dsc$w_length],
1153 1245 2 | .desc[dsc$a_pointer],
1154 1246 2 | lffill)
1155 1247 2 | THEN
1156 1248 2 | BEGIN
1157 1249 2 | SIGNAL(set$_invquaval, 2, desc, SD_LFFILL);
1158 1250 2 | RETURN 0;
1159 1251 2 | END;
1160 1252 2 | END;
1161 1253 2 | IF .lffill LSS 0
1162 1254 2 | OR .lffill GTR 9
1163 1255 2 |
```

```
1164 1256 3 THEN
1165 1257 4 BEGIN
1166 1258 4 SIGNAL(set$_invquaval, 2, desc, SD_LFFILL);
1167 1259 4 RETURN 0;
1168 1260 3 END;
1169 1261 3 IF .lffill EQL 0
1170 1262 3 THEN tt1_clr = .tt1_clr OR tt$m_lffill
1171 1263 3 ELSE tt1_set = .tt1_set OR tt$m_lffill;
1172 1264 3 fill<8,8> = .lffill
1173 1265 2 END;
1174 1266 2
1175 1267 2
1176 1268 2 When storing the fill characteristics above, it is important that
1177 1269 2 changing one doesn't zero out the other, because either or both
1178 1270 2 may have previously been set (by get_term_def).
1179 1271 2
1180 1272 2
1181 1273 2
1182 1274 2 Speed.
1183 1275 2
1184 1276 2 IF cli$present(SD_SPEED)
1185 1277 2 THEN
1186 1278 2 BEGIN
1187 1279 2 IF .flags[set$_perm]
1188 1280 2 THEN tt2_clr = .tt2_clr OR tt2$m_autobaud;
1189 1281 2 flags[set$_speed] = 1;
1190 1282 2 speed = 0;
1191 1283 2 INCR j FROM 0 TO 1 DO
1192 1284 2 BEGIN
1193 1285 2 IF cli$get_value(SD_SPEED, desc)
1194 1286 2 THEN INCR i FROM 0 TO term$_spdnum-1 DO
1195 1287 2 BEGIN
1196 1288 2 BIND spd_desc = term$_spdblck[i] : REF VECTOR;
1197 1289 2 IF CH$EQ[.desc[dsc$_length], .desc[dsc$_pointer],
1198 1290 2 .desc[dsc$_length], .spd_desc[1])
1199 1291 2 THEN (speed = .speed*8 OR .i; EXITLOOP);
1200 1292 2 END;
1201 1293 2 END;
1202 1294 2 END;
1203 1295 2
1204 1296 2
1205 1297 2 There are a number of different keywords which cause bits in the
1206 1298 2 two characteristics longwords to be set. This is relatively simple,
1207 1299 2 since most keywords correspond to the setting of a corresponding bit,
1208 1300 2 and the negation of a keyword causes the bit to be cleared. Deal with
1209 1301 2 those here.
1210 1302 2 -Exception: The qualifier /DEC_CRT does not uniquely correspond with
1211 1303 2 a characteristic. Instead it can have two values, 1 or 2
1212 1304 2 which can modify the characteristics, decrt and decrt2.
1213 1305 2 Therefore this qualifier is a special case which is proc-
1214 1306 2 essed further on in this routine.
1215 1307 2
1216 1308 2 INCR i FROM 0 TO term$_ttset_num-1 DO ! 1st char longword
1217 1309 2 BEGIN
1218 1310 2 status = cli$present(.term$_ttset_key[i]);
1219 1311 2 IF .status
1220 1312 2 THEN tt1_set = .tt1_set OR .term$_ttset_bit[i]
```



```
1221 1313 ELSE IF .status EQL cli$ negated
1222 1314 THEN tt1_clr = .tt1_clr OR .term$_ttset_bit[i];
1223 1315 END;
1224 1316 INCR i FROM 0 TO term$_tt2set_num-1 DO ! 2nd char longword
1225 1317 BEGIN
1226 1318 status = cli$present(.term$_tt2set_key[i]);
1227 1319 IF .status
1228 1320 THEN tt2_set = .tt2_set OR .term$_tt2set_bit[i]
1229 1321 ELSE IF .status EQL cli$ negated
1230 1322 THEN tt2_clr = .tt2_clr OR .term$_tt2set_bit[i];
1231 1323 END;
1232 1324
1233 1325
1234 1326 ! Now for the corkers. Some (6 at present) keywords cause bits
1235 1327 to be cleared; and the negation of those keywords causes the
1236 1328 bit to be set. So...
1237 1329
1238 1330 INCR i FROM 0 to term$_ttclr_num-1 DO
1239 1331 BEGIN
1240 1332 status = cli$present(.term$_ttclr_key[i]);
1241 1333 IF .status
1242 1334 THEN tt1_clr = .tt1_clr OR .term$_ttclr_bit[i]
1243 1335 ELSE IF .status EQL cli$ negated
1244 1336 THEN tt1_set = .tt1_set OR .term$_ttclr_bit[i];
1245 1337 END;
1246 1338 INCR i FROM 0 to term$_tt2clr_num-1 DO
1247 1339 BEGIN
1248 1340 status = cli$present(.term$_tt2clr_key[i]);
1249 1341 IF .status
1250 1342 THEN tt2_clr = .tt2_clr OR .term$_tt2clr_bit[i]
1251 1343 ELSE IF .status EQL cli$ negated
1252 1344 THEN tt2_set = .tt2_set OR .term$_tt2clr_bit[i];
1253 1345 END;
1254 1346
1255 1347 ! LOCAL ECHO IMPLIES NOECHO
1256 1348
1257 1349 status = cli$present(%ASCII 'LOCAL_ECHO');
1258 1350 IF .status
1259 1351 THEN tt1_set = .tt1_set OR tt$m_noecho
1260 1352 ELSE IF .status EQL cli$ negated
1261 1353 THEN tt1_clr = .tt1_clr OR tt$m_noecho;
1262 1354
1263 1355
1264 1356 ! Dismiss parity error modifier has a bit set in the parity longword
1265 1357
1266 1358 status = cli$present(%ASCII 'DISMISS_PARITY');
1267 1359 IF .status
1268 1360 THEN
1269 1361 BEGIN
1270 1362 flags[set$v_dismiss] = 1;
1271 1363 parity = .parity OR tt$m_altdispar OR tt$m_disparerr;
1272 1364 END;
1273 1365 IF .status EQL cli$ negated
1274 1366 THEN
1275 1367 BEGIN
1276 1368 flags[set$v_nodism] = 1;
1277 1369 parity = .parity OR tt$m_altdispar;
```

```
1278 1370 2 END;
1279 1371 2
1280 1372 2
1281 1373 2 Qualifier - / [NO]DEC CRT[=number]
1282 1374 2 This qualifier can set on or off the characteristics, DEC CRT and DEC CRT2
1283 1375 2 If the number specified is 1 (also the default value) the characteristic to
1284 1376 2 modify is dec_crt. If the number is 2 then the characteristic is DEC CRT2.
1285 1377 2
1286 1378 2 status = cli$present( SD DEC CRT) ;
1287 1379 2 IF (.status OR (.status EQL cli$_negated)) THEN
1288 1380 2 BEGIN
1289 1381 2 status2 = cli$get_value(SD_DEC_CRT, desc) ;
1290 1382 2 INCR J FROM 0 TO T DO
1291 1383 2 BEGIN
1292 1384 2 IF NOT .status2 THEN
1293 1385 2 decrt = 1
1294 1386 2 ELSE
1295 1387 2 BEGIN
1296 1388 2 IF NOT lib$cvd_dtb(.desc[dsc$w_length],
1297 1389 2 .desc[dsc$a_pointer],
1298 1390 2 decrt)
1299 1391 2 THEN
1300 1392 2 BEGIN
1301 1393 2 SIGNAL(set$_invquaval, 2, desc, SD_DEC_CRT);
1302 1394 2 RETURN 0;
1303 1395 2 END;
1304 1396 2 END;
1305 1397 2 IF ((.decrt NEQ 1) AND (.decrt NEQ 2)) THEN
1306 1398 2 BEGIN
1307 1399 2 SIGNAL(set$_invquaval, 2, desc, SD_DEC_CRT);
1308 1400 2 RETURN 0;
1309 1401 2 END;
1310 1402 2 IF .status THEN
1311 1403 2 BEGIN
1312 1404 2 IF .decrt EQL 1 THEN
1313 1405 2 decrt_set = .decrt_set OR tt2$m_decrt OR tt2$m_ansi
1314 1406 2 ELSE
1315 1407 2 decrt_set = .decrt_set OR tt2$m_decrt2 OR tt2$m_decrt OR tt2$m_ansi
1316 1408 2 END;
1317 1409 2 IF .status EQL cli$_negated THEN
1318 1410 2 BEGIN
1319 1411 2 IF .decrt EQL 1 THEN
1320 1412 2 decrt_clr = .decrt_clr OR tt2$m_decrt OR tt2$m_decrt2
1321 1413 2 ELSE
1322 1414 2 decrt_clr = .decrt_clr OR tt2$m_decrt2;
1323 1415 2 END;
1324 1416 2 status2 = cli$get_value(SD_DEC_CRT, desc) ;
1325 1417 2 IF NOT .status2 THEN exitloop ;
1326 1418 2 END ;
1327 1419 2 END ;
1328 1420 2
1329 1421 2
1330 1422 2 One more special condition. If ansi was turned off then
1331 1423 2 decrt levels 1 and 2 must also be turned off since they are
1332 1424 2 a superset of ansi.
1333 1425 2
1334 1426 2 status = cli$present(%ASCII 'ANSI_CRT') ;
```



```
: 1335      1427 2 IF .status EQL cli$ negated THEN
: 1336      1428 2     tt2_clr = .tt2_clr OR tt2$m_deccrt OR tt2$m_deccrt2 ;
: 1337      1429 2
: 1338      1430 2
: 1339      1431 2     ! Now set/clear the bits, using STATUS as a mask longword.
: 1340      1432 2
: 1341      1433 2     status = .tt1_set OR .tt1_clr ;
: 1342      1434 2     info_block[term$l_set1] = .info_block[term$l_set1]
: 1343      1435 2         AND NOT .status
: 1344      1436 2         OR .tt1_set ;
: 1345      1437 2     status = .tt2_set OR .tt2_clr OR .deccrt_set OR .deccrt_clr ;
: 1346      1438 2     info_block[term$l_set2] = .info_block[term$l_set2]
: 1347      1439 2         AND NOT .status
: 1348      1440 2         OR .tt2_set
: 1349      1441 2         OR .deccrt_set ;
: 1350      1442 2
: 1351      1443 2
: 1352      1444 2     ! If /AUTOBAUD was specified, and no speed, then set the speed to 9600
: 1353      1445 2
: 1354      1446 2     IF (.tt2_set AND tt2$m_autobaud) NEQ 0          ! If /AUTOBAUD specified
: 1355      1447 2     AND .speed EQL 0                               ! and no speed set
: 1356      1448 2     THEN speed = tt$c_baud_9600;
: 1357      1449 2
: 1358      1450 2 RETURN 1;
: 1359      1451 1 END;
```

```
                                .PSECT $SPLITS$,NOWRT,NOEXE,2
                                00 00 4F 48 43 45 5F 4C 41 43 4F 4C 00190 P.ABT: .ASCII \LOCAL ECHO\<0><0>
                                010E000A 0019C P.ABS: .LONG 17694730
                                00000000' 001A0 .ADDRESS P.ABT
00 59 54 49 52 41 50 5F 53 53 49 4D 53 49 44 001A4 P.ABV: .ASCII \DISMISS_PARITY\<0><0>
                                00 001B3
                                010E000E 001B4 P.ABU: .LONG 17694734
                                00000000' 001B8 .ADDRESS P.ABV
                                54 52 43 5F 49 53 4E 41 001BC P.ABX: .ASCII \ANSI CRT\
                                010E0008 001C4 P.ABW: .LONG 17694728
                                00000000' 001C8 .ADDRESS P.ABX
```

```
                                .PSECT $CODE$,NOWRT,2
                                OFFC 00000 GET_VALUES:
                                .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
                                5E      24 C2 00002      SUBL2 #36, SP
                                54      04 AC D0 00005      MOVL DATA_BUFFER, R4
                                5A      04 A4 9E 00009      MOVAB 4(R4), R10
                                08      A4 9F 0000D      PUSHAB 8(R4)
                                5B      0C A4 9E 00010      MOVAB 12(R4), R11
                                10      A4 9F 00014      PUSHAB 16(R4)
                                56      14 A4 9E 00017      MOVAB 20(R4), R6
                                58      1C A4 9E 0001B      MOVAB 28(R4), R8
                                57      28 A4 9E 0001F      MOVAB 40(R4), R7
                                3C      A4 9F 00023      PUSHAB 60(R4)
```

: 1028
: 1052
:

	2C	AE	020E0000	40	A4	9F	00026	PUSHAB	64(R4)		
				30	8F	D0	00029	MOVL	#34471936, DESC		1067
			0000'		AE	D4	00031	CLRL	DESC+4		
00000000G	00				CF	9F	00034	PUSHAB	SD_PARITY		1072
	59				01	FB	00038	CALLS	#1, CLISPRESENT		
	60				50	D0	0003F	MOVL	R0, STATUS		
	66		60		59	E9	00042	BLBC	STATUS, 4\$		1073
			2C		8F	88	00045	BISB2	#96, (R6)		1076
			0000'		AE	9F	00049	PUSHAB	DESC		1077
					CF	9F	0004C	PUSHAB	SD_PARITY		
00000000G	00				02	FB	00050	CALLS	#2, CLISGET_VALUE		
	4B				50	E9	00057	BLBC	R0, 4\$		
	55		2C		AE	3C	0005A	MOVZWL	DESC, R5		1080
	50	00000000G			00	D0	0005E	MOVL	TERMS_ODD+4, R0		1081
60	30	BE			55	29	00065	CMPC3	R5, @DESC+4, (R0)		1080
					09	12	0006A	BNEQ	1\$		
	66		80		8F	88	0006C	BISB2	#128, (R6)		1084
	68				04	88	00070	BISB2	#4, (R8)		1085
					30	11	00073	BRB	4\$		1080
	50	00000000G			00	D0	00075	MOVL	TERMS_EVEN+4, R0		1088
60	30	BE			55	29	0007C	CMPC3	R5, @DESC+4, (R0)		1087
					05	12	00081	BNEQ	2\$		
	68				08	88	00083	BISB2	#8, (R8)		1089
					1D	11	00086	BRB	4\$		
	50	00000000G			00	D0	00088	MOVL	TERMS_NONE+4, R0		1091
60	30	BE			55	29	0008F	CMPC3	R5, @DESC+4, (R0)		1090
					09	12	00094	BNEQ	3\$		
	59	00000000G			8F	D0	00096	MOVL	#CLIS_NEGATED, STATUS		1092
			0000'		06	11	0009D	BRB	4\$		
					CF	9F	0009F	PUSHAB	SD_PARITY		1095
					5F	11	000A3	BRB	7\$		
00000000G	8F				59	D1	000A5	CMPL	STATUS, #CLIS_NEGATED		1100
					0F	12	000AC	BNEQ	5\$		
50	66	0000000C0			8F	CB	000AE	BICL3	#192, (R6), R0		1103
66	50				20	C9	000B6	BISL3	#32, R0, (R6)		
	68				10	88	000BA	BISB2	#16, (R8)		1104
			0000'		CF	9F	000BD	PUSHAB	SD_FRAME		1109
00000000G	00				01	FB	000C1	CALLS	#1, CLISPRESENT		
	49				50	E9	000C8	BLBC	R0, 9\$		
			2C		AE	9F	000CB	PUSHAB	DESC		1113
			0000'		CF	9F	000CE	PUSHAB	SD_FRAME		
00000000G	00				02	FB	000D2	CALLS	#2, CLISGET_VALUE		
	2A				50	E9	000D9	BLBC	R0, 8\$		
			14		AE	9F	000DC	PUSHAB	FRAME		1116
			34		AE	DD	000DF	PUSHL	DESC+4		1117
			34		AE	3C	000E2	MOVZWL	DESC, -(SP)		1116
00000000G	00				03	FB	000E6	CALLS	#3, LIB\$CVT_DTB		
	10				50	E9	000ED	BLBC	R0, 6\$		
	50		14		AE	D0	000F0	MOVL	FRAME, R0		1124
					10	13	000F4	BEQL	8\$		
	05				50	D1	000F6	CMPL	R0, #5		1125
					05	19	000F9	BLSS	6\$		
	08				50	D1	000FB	CMPL	R0, #8		1126
					06	15	000FE	BLEQ	8\$		
			0000'		CF	9F	00100	PUSHAB	SD_FRAME		1129
					5B	11	00104	BRB	11\$		
66	56				10	C9	00106	BISL3	#16, R6, (R6)		1133

66	04	00	14	AE	FO	0010A	INSV	FRAME, #0, #4, (R6)	1134
	01	A8		04	88	00110	BISB2	#4, 1(R8)	1135
			0000'	CF	9F	00114	PUSHAB	SD_PAGE	1140
	00000000G	00		01	FB	00118	CALLS	#1, CLISPRESNT	
		46		50	E9	0011F	BLBC	R0, 13\$	
	01	A8		02	88	00122	BISB2	#2, 1(R8)	1144
			18	AE	D4	00126	CLRL	PAGE	1145
			2C	AE	9F	00129	PUSHAB	DESC	1146
			0000'	CF	9F	0012C	PUSHAB	SD_PAGE	
	00000000G	00		02	FB	00130	CALLS	#2, CLISGET_VALUE	
		29		50	E9	00137	BLBC	R0, 12\$	
			18	AE	9F	0013A	PUSHAB	PAGE	1149
			34	AE	DD	0013D	PUSHL	DESC+4	1150
		7E	34	AE	3C	00140	MOVZWL	DESC, -(SP)	1149
	00000000G	00		03	FB	00144	CALLS	#3, LIB\$CVT_DTB	
		0F		50	E9	0014B	BLBC	R0, 10\$	
			18	AE	D5	0014E	TSTL	PAGE	1157
				0A	19	00151	BLSS	10\$	
	000000FF	8F	18	AE	D1	00153	CMPL	PAGE, #255	1158
				06	15	0015B	BLEQ	12\$	
			0000'	CF	9F	0015D	PUSHAB	SD_PAGE	1161
				52	11	00161	BRB	15\$	
	07	A7	18	AE	90	00163	MOVB	PAGE, 7(R7)	1165
			0000'	CF	9F	00168	PUSHAB	SD_WIDTH	1171
	00000000G	00		01	FB	0016C	CALLS	#1, CLISPRESNT	
		46		50	E9	00173	BLBC	R0, 17\$	
	01	A8		01	88	00176	BISB2	#1, 1(R8)	1175
			1C	AE	D4	0017A	CLRL	WIDTH	1176
			2C	AE	9F	0017D	PUSHAB	DESC	1177
			0000'	CF	9F	00180	PUSHAB	SD_WIDTH	
	00000000G	00		02	FB	00184	CALLS	#2, CLISGET_VALUE	
		29		50	E9	0018B	BLBC	R0, 16\$	
			1C	AE	9F	0018E	PUSHAB	WIDTH	1180
			34	AE	DD	00191	PUSHL	DESC+4	1181
		7E	34	AE	3C	00194	MOVZWL	DESC, -(SP)	1180
	00000000G	00		03	FB	00198	CALLS	#3, LIB\$CVT_DTB	
		0F		50	E9	0019F	BLBC	R0, 14\$	
			1C	AE	D5	001A2	TSTL	WIDTH	1188
				0A	19	001A5	BLSS	14\$	
	000001FF	8F	1C	AE	D1	001A7	CMPL	WIDTH, #511	1189
				06	15	001AF	BLEQ	16\$	
			0000'	CF	9F	001B1	PUSHAB	SD_WIDTH	1192
				4E	11	001B5	BRB	20\$	
	02	A7	1C	AE	B0	001B7	MCVW	WIDTH, 2(R7)	1196
			20	AE	D4	001BC	CLRL	CRFILL	1202
			0000'	CF	9F	001BF	PUSHAB	SD_CRFILL	1204
	00000000G	00		01	FB	001C3	CALLS	#1, CLISPRESNT	
		4C		50	E9	001CA	BLBC	R0, 24\$	
		68	40	8F	88	001CD	BISB2	#64, (R8)	1207
			2C	AE	9F	001D1	PUSHAB	DESC	1208
			0000'	CF	9F	001D4	PUSHAB	SD_CRFILL	
	00000000G	00		02	FB	001D8	CALLS	#2, CLISGET_VALUE	
		14		50	E9	001DF	BLBC	R0, 18\$	
			20	AE	9F	001E2	PUSHAB	CRFILL	1211
			34	AE	DD	001E5	PUSHL	DESC+4	1212
		7E	34	AE	3C	001E8	MOVZWL	DESC, -(SP)	1211
	00000000G	00		03	FB	001EC	CALLS	#3, LIB\$CVT_DTB	

	0B		50	E9	001F3	BLBC	R0, 19\$			
	52	20	AE	D0	001F6	18\$:	MOVL	CRFILL, R2	1220	
			05	19	001FA		BLSS	19\$		
	09		52	D1	001FC		CMPL	R2, #9	1221	
			06	15	001FF		BLEQ	21\$		
		0000'	CF	9F	00201	19\$:	PUSHAB	SD CRFILL	1224	
			5A	11	00205	20\$:	BRB	27\$		
			52	D5	00207	21\$:	TSTL	R2	1227	
			06	12	00209		BNEQ	22\$		
01	AA		04	88	0020B		BISB2	#4, 1(R10)	1228	
			04	11	0020F		BRB	23\$		
01	A4		04	88	00211	22\$:	BISB2	#4, 1(R4)	1229	
18	A4		52	90	00215	23\$:	MOVB	R2, 24(R4)	1230	
		24	AE	D4	00219	24\$:	CLRL	LFFILL	1236	
		0000'	CF	9F	0021C		PUSHAB	SD LFFILL	1238	
00000000G	00		01	FB	00220		CALLS	#1, CLISPRESENT		
	4C		50	E9	00227		BLBC	R0, 31\$		
	68		20	88	0022A		BISB2	#32, (R8)	1241	
		2C	AE	9F	0022D		PUSHAB	DESC	1242	
		0000'	CF	9F	00230		PUSHAB	SD LFFILL		
00000000G	00		02	FB	00234		CALLS	#2, CLISGET_VALUE		
	14		50	E9	0023B		BLBC	R0, 25\$		
		24	AE	9F	0023E		PUSHAB	LFFILL	1245	
		34	AE	DD	00241		PUSHL	DESC+4	1246	
		34	AE	3C	00244		MOVZWL	DESC, -(SP)	1245	
00000000G	7E		03	FB	00248		CALLS	#3, LIBSCVT_DTB		
	00		50	E9	0024F		BLBC	R0, 26\$		
	0B		24	AE	D0	00252	25\$:	MOVL	LFFILL, R2	1254
	52		05	19	00256		BLSS	26\$		
	09		52	D1	00258		CMPL	R2, #9	1255	
			07	15	0025B		BLEQ	28\$		
		0000'	CF	9F	0025D	26\$:	PUSHAB	SD LFFILL	1258	
			020F	31	00261	27\$:	BRW	59\$		
			52	D5	00264	28\$:	TSTL	R2	1261	
			06	12	00266		BNEQ	29\$		
01	AA		08	88	00268		BISB2	#8, 1(R10)	1262	
			04	11	0026C		BRB	30\$		
01	A4		08	88	0026E	29\$:	BISB2	#8, 1(R4)	1263	
19	A4		52	90	00272	30\$:	MOVB	R2, 25(R4)	1264	
		0000'	CF	9F	00276	31\$:	PUSHAB	SD SPEED	1276	
00000000G	00		01	FB	0027A		CALLS	#1, CLISPRESENT		
	51		50	E9	00281		BLBC	R0, 37\$		
03	68		01	E1	00284		BBC	#1, (R8), 32\$	1279	
	68		02	88	00288		BISB2	#2, (R11)	1280	
	68		8F	88	0028B	32\$:	BISB2	#128, (R8)	1281	
		80	BE	D4	0028F		CLRL	@8(SP)	1282	
		08	AE	D4	00292		CLRL	J	1283	
		10	AE	9F	00295	33\$:	PUSHAB	DESC	1285	
		2C	CF	9F	00298		PUSHAB	SD SPEED		
		0000'	02	FB	0029C		CALLS	#2, CLISGET_VALUE		
00000000G	00		50	E9	002A3		BLBC	R0, 36\$		
	2A		01	CE	002A6		MNEGL	#1, 1	1289	
	55		1D	11	002A9		BRB	35\$		
		00000000G	045	D0	002AB	34\$:	MOVL	TERMS_SPDBLK[1], R0	1290	
04	B0	30	BE	29	002B3		CMPC3	DESC, @DESC+4, @4(R0)	1289	
			0C	12	002BA		BNEQ	35\$		
	50	08	BE	08	78	002BC	ASHL	#8, @8(SP), R0	1291	

08	BE	50	55	C9	002C1	BISL3	I, R0, @8(SP)	
	DB		08	11	002C6	BRB	36\$	
	CO	10	55	F3	002C8	AOBLEQ	#TERMS_SPDNUM-1, I, 34\$	1286
			AF	01	002D0	AOBLEQ	#1, J, 33\$	1283
			52	01	002D5	MNEGL	#1, I	1308
				2F	11	002D8	BRB	40\$
		00000000G	00	DD	002DA	PUSHL	TERMS_TTSET_KEY[I]	1310
			59	01	FB	002E1	CALLS	#1, C[ISPRESENT
			0A	50	D0	002E8	MOVL	R0, STATUS
			64	59	E9	002EB	BLBC	STATUS, 39\$
		00000000G	00	C8	002EE	BISL2	TERMS_TTSET_BIT[I], (R4)	1311
				11	11	002F6	BRB	40\$
		00000000G	8F	D1	002F8	CMPL	STATUS, #CLIS_NEGATED	1312
				08	12	002FF	BNEQ	40\$
		00000000G	6A	C8	00301	BISL2	TERMS_TTSET_BIT[I], (R10)	1313
C9			52	F3	00309	AOBLEQ	#TERMS_TTSET_NUM-1, I, 38\$	1314
			52	01	00311	MNEGL	#1, I	1308
				30	11	00314	BRB	43\$
		00000000G	00	DD	00316	PUSHL	TERMS_TT2SET_KEY[I]	1316
			59	01	FB	0031D	CALLS	#1, C[ISPRESENT
			0B	50	D0	00324	MOVL	R0, STATUS
		0C	BE	59	E9	00327	BLBC	STATUS, 42\$
				C8	0032A	BISL2	TERMS_TT2SET_BIT[I], @12(SP)	1319
				11	11	00333	BRB	43\$
		00000000G	8F	D1	00335	CMPL	STATUS, #CLIS_NEGATED	1320
				08	12	0033C	BNEQ	43\$
		00000000G	6B	C8	0033E	BISL2	TERMS_TT2SET_BIT[I], (R11)	1321
C8			52	F3	00346	AOBLEQ	#TERMS_TT2SET_NUM-1, I, 41\$	1322
			52	01	0034E	MNEGL	#1, I	1316
				2F	11	00351	BRB	46\$
		00000000G	00	DD	00353	PUSHL	TERMS_TTCLR_KEY[I]	1330
			59	01	FB	0035A	CALLS	#1, C[ISPRESENT
			0A	50	D0	00361	MOVL	R0, STATUS
			6A	59	E9	00364	BLBC	STATUS, 45\$
		00000000G	00	C8	00367	BISL2	TERMS_TTCLR_BIT[I], (R10)	1333
				11	11	0036F	BRB	46\$
		00000000G	8F	D1	00371	CMPL	STATUS, #CLIS_NEGATED	1334
				08	12	00378	BNEQ	46\$
		00000000G	64	C8	0037A	BISL2	TERMS_TTCLR_BIT[I], (R4)	1335
C9			52	F3	00382	AOBLEQ	#TERMS_TTCLR_NUM-1, I, 44\$	1336
			52	01	0038A	MNEGL	#1, I	1330
				30	11	0038D	BRB	49\$
		00000000G	00	DD	0038F	PUSHL	TERMS_TT2CLR_KEY[I]	1338
			59	01	FB	00396	CALLS	#1, C[ISPRESENT
			0A	50	D0	0039D	MOVL	R0, STATUS
			6B	59	E9	003A0	BLBC	STATUS, 48\$
		00000000G	00	C8	003A3	BISL2	TERMS_TT2CLR_BIT[I], (R11)	1341
				12	11	003AB	BRB	49\$
		00000000G	8F	D1	003AD	CMPL	STATUS, #CLIS_NEGATED	1342
				09	12	003B4	BNEQ	49\$
		0C	BE	C8	003B6	BISL2	TERMS_TT2CLR_BIT[I], @12(SP)	1343
C8			52	F3	003BF	AOBLEQ	#TERMS_TT2CLR_NUM-1, I, 47\$	1344
				CF	9F	003C7	PUSHL	P.ABS
		00000000G	00	01	FB	003CB	CALLS	#1, C[ISPRESENT
			59	50	D0	003D2	MOVL	R0, STATUS
			05	59	E9	003D5	BLBC	STATUS, 50\$
			64	02	88	003D8	BISB2	#2, (R4)

00000000G	8F		0C	11	003DB	BRB	51\$		
			59	D1	003DD	CMPL	STATUS, #CLIS_NEGATED		1352
	6A		03	12	003E4	BNEQ	51\$		
		0000'	02	88	003E6	BISB2	#2, (R10)		1353
00000000G	00		CF	9F	003E9	PUSHAB	P.ABU		1358
	59		01	FB	003ED	CALLS	#1, CLISPRESENT		
	08		50	D0	003F4	MOVL	R0, STATUS		
	01		59	E9	003F7	BLBC	STATUS, 52\$		1359
	01		08	88	003FA	BISB2	#8, 1(R8)		1362
	01		06	88	003FE	BISB2	#6, 1(R6)		1363
00000000G	8F		59	D1	00402	CMPL	STATUS, #CLIS_NEGATED		1365
			08	12	00409	BNEQ	53\$		
	01		10	88	0040B	BISB2	#16, 1(R8)		1368
	01		04	88	0040F	BISB2	#4, 1(R6)		1369
		0000'	CF	9F	00413	PUSHAB	SD_DEC CRT		1378
00000000G	00		01	FB	00417	CALLS	#1, CLISPRESENT		
	59		50	D0	0041E	MOVL	R0, STATUS		
	0C		59	E8	00421	BLBS	STATUS, 54\$		1379
00000000G	8F		59	D1	00424	CMPL	STATUS, #CLIS_NEGATED		
			03	13	0042B	BEQL	54\$		
		00AD	31	0042D	BRW	65\$			
		2C	AE	9F	00430	PUSHAB	DESC		1381
		0000'	CF	9F	00433	PUSHAB	SD_DEC CRT		
00000000G	00		02	FB	00437	CALLS	#2, CLISGET_VALUE		
	55		50	D0	0043E	MOVL	R0, STATUS2		
	52		55	D2	00441	MCOML	STATUS2, R2		1384
			53	D4	00444	CLRL	J		
	06		52	E9	00446	BLBC	R2, 56\$		
28	AE		01	D0	00449	MOVL	#1, DECCRT		1385
			14	11	0044D	BRB	57\$		
		28	AE	9F	0044F	PUSHAB	DECCRT		1388
		34	AE	DD	00452	PUSHL	DESC+4		1389
		34	AE	3C	00455	MOVZWL	DESC, -(SP)		1388
00000000G	00		03	FB	00459	CALLS	#3, LIB\$CVT_DTB		
	0C		50	E9	00460	BLBC	R0, 58\$		
	01	28	AE	D1	00463	CMPL	DECCRT, #1		1397
			1F	13	00467	BEQL	60\$		
	02	28	AE	D1	00469	CMPL	DECCRT, #2		
			19	13	0046D	BEQL	60\$		
		0000'	CF	9F	0046F	PUSHAB	SD_DEC CRT		1399
		30	AE	9F	00473	PUSHAB	DESC		
			02	DD	00476	PUSHL	#2		
		0077132A	8F	DD	00478	PUSHL	#7803690		
00000000G	00		04	FB	0047E	CALLS	#4, LIB\$SIGNAL		
			00AE	31	00485	BRW	68\$		1400
	18		59	E9	00488	BLBC	STATUS, 62\$		1402
	01	28	AE	D1	0048B	CMPL	DECCRT, #1		1404
			0A	12	0048F	BNEQ	61\$		
	04	BE 21000000	8F	C8	00491	BISL2	#553648128, @4(SP)		1405
			08	11	00499	BRB	62\$		
	04	BE 61000000	8F	C8	0049B	BISL2	#1627389952, @4(SP)		1407
00000000G	8F		59	D1	004A3	CMPL	STATUS, #CLIS_NEGATED		1409
			14	12	004AA	BNEQ	64\$		
	01	28	AE	D1	004AC	CMPL	DECCRT, #1		1411
			08	12	004B0	BNEQ	63\$		
00	BE	02	03	F0	004B2	INSV	#3, #29, #2, @0(SP)		1412
			06	11	004B8	BRB	64\$		

SETTERM
V04-000

D 13
16-Sep-1984 01:10:06
14-Sep-1984 12:09:20

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETTERM.B32;1

Page 43
(7)

00	BE	01	1E		01	F0	004BA	63\$:	INSV	#1, #30, #1, @0(SP)	1414
				2C	AE	9F	004C0	64\$:	PUSHAB	DESC	1416
				0000'	CF	9F	004C3		PUSHAB	SD_DEC CRT	
		00000000G	00		02	FB	004C7		CALLS	#2, CLISGET_VALUE	
			55		50	D0	004CE		MOVL	R0, STATUS2	
			52		55	D2	004D1		MCOML	STATUS2, R2	1417
			06		52	E8	004D4		BLBS	R2, 65\$	
FF69		53	01		01	F1	004D7		ACBL	#1, #1, J, 55\$	1382
				0000'	CF	9F	004DD	65\$:	PUSHAB	P.ABW	1426
		00000000G	00		01	FB	004E1		CALLS	#1, CLISPRESENT	
			59		50	D0	004E8		MOVL	R0, STATUS	
		00000000G	8F		59	D1	004EB		CMPL	STATUS, #CLIS_NEGATED	1427
					05	12	004F2		BNEQ	66\$	
		03	AB	60	8F	88	004F4		BISB2	#96, 3(R11)	1428
	59		64		6A	C9	004F9	66\$:	BISL3	(R10), (R4), STATUS	1433
	50	04	A7		59	CB	004FD		BICL3	STATUS, 4(R7), R0	1435
04	A7		50		64	C9	00502		BISL3	(R4), R0, 4(R7)	1436
	50	0C	BE		6B	C9	00507		BISL3	(R11), @12(SP), R0	1437
			50	04	BE	C8	0050C		BISL2	@4(SP), R0	
	59		50	00	BE	C9	00510		BISL3	@0(SP), R0, STATUS	
	50	08	A7		59	CB	00515		BICL3	STATUS, 8(R7), R0	1439
			50	0C	BE	C8	0051A		BISL2	@12(SP), R0	1440
08	A7		50	04	BE	C9	0051E		BISL3	@4(SP), R0, 8(R7)	1441
	09	0C	BE		01	E1	00524		BBC	#1, @12(SP), 67\$	1446
				08	BE	D5	00529		TSTL	@8(SP)	1447
					04	12	0052C		BNEQ	67\$	
		08	BE		0F	D0	0052E		MOVL	#15, @8(SP)	1448
			50		01	D0	00532	67\$:	MOVL	#1, R0	1450
						04	00535		RET		
					50	D4	00536	68\$:	CLRL	R0	1451
					04	00538			RET		

; Routine Size: 1337 bytes, Routine Base: \$CODE\$ + 0549

```
1361 1452 1 ROUTINE inquire_type (data_buffer) =
1362 1453 2 BEGIN
1363 1454 3
1364 1455 4 ++
1365 1456 5 Functional description
1366 1457 6
1367 1458 7 This routine implements the SET TERM/INQUIRE qualifier. It sends
1368 1459 8 a message to the terminal, saying "who are you?" If the terminal
1369 1460 9 responds, then we try to find a match between the pattern given and
1370 1461 10 the known patterns that supported DEC terminals give.
1371 1462 11
1372 1463 12 Inputs
1373 1464 13 INFO_BLOCK - the characteristics buffer.
1374 1465 14 INDEX - number which points to terminal's terminal block
1375 1466 15
1376 1467 16 Outputs
1377 1468 17 INFO_BLOCK - will be changed according to what device type was
1378 1469 18 INDEX - will be a number, which will index to the terminal
1379 1470 19
1380 1471 20 ----
1381 1472 21
1382 1473 22 MAP
1383 1474 23 data_buffer : REF VECTOR;
1384 1475 24
1385 1476 25 Bind the data buffer to names we understand.
1386 1477 26
1387 1478 27 bind_data;
1388 1479 28
1389 1480 29 LOCAL
1390 1481 30 ptr, ! Pointer to end of answer
1391 1482 31 resp_len, ! Answer length
1392 1483 32 sequence : REF VECTOR[BYTE], ! Real answer
1393 1484 33 resp_buffer : VECTOR[80,BYTE], ! Place to put answer
1394 1485 34 INITIAL(REF 20 OF (0)), ! Zero it out first
1395 1486 35 iosb : VECTOR[4,WORD], ! I/O status block
1396 1487 36 tmpblock : $bblock[12],
1397 1488 37 status;
1398 1489 38
1399 1490 39
1400 1491 40 There are a number of different request strings, strings which will cause
1401 1492 41 different terminals to respond with their identification string. Loop
1402 1493 42 thru these different strings, seeing if the terminal responds to any of
1403 1494 43 them.
1404 1495 44
1405 1496 45 INCR i FROM 0 to term$_reqnum - 1 DO
1406 1497 46 BEGIN
1407 1498 47 BIND request = .term$_reqblk[i] : VECTOR;
1408 1499 48 status = $QIOW(FUNC = io$_readprompt OR io$_m_timed OR io$_m_purge OR io$_m_escape OR io$_m_noecho,
P 1500 49 CHAN = .chan,
P 1501 50 IOSB = iosb,
P 1502 51 P1 = resp_buffer, ! Place to store response
P 1503 52 P2 = %ALLOCATION(resp_buffer), ! How big is the buffer
P 1504 53 P3 = 4, ! Wait several seconds
P 1505 54 P5 = .request[1], ! Address of request string,
P 1506 55 P6 = .request[0]); ! Length of request string,
1416 1507 56 IF .status ! Check both statuses
1417 1508 57 THEN status = .iosb[0];
```



```
1418 1509 3 IF (NOT .status) AND
1419 1510 3 (.status NEQ ss$_timeout) AND
1420 1511 4 (.status NEQ ss$_badescape)
1421 1512 3 THEN
1422 1513 4 BEGIN
1423 1514 4 SIGNAL(set$_writeerr AND NOT sts$_severity OR sts$_warning,
1424 1515 4 1, dev_desc,
1425 1516 4 .status);
1426 1517 4 RETURN 0;
1427 1518 3 END;
1428 1519 3 IF .i EQL 1 THEN
1429 1520 4 BEGIN
1430 1521 4 tmpblock = .info_block;
1431 1522 4 tmpblock+4 = (.info_block+4) OR tt$_eightbit;
1432 1523 4 $qio (chan = .chan,
1433 1524 4 func = io$_setmode,
1434 1525 4 p1 = tmpblock);
1435 1526 3 END;
1436 1527 3 IF .status NEQ ss$_timeout
1437 1528 3 THEN
1438 1529 4 Begin
1439 1530 4
1440 1531 4
1441 1532 4 At this point, we at least have something. IOSB[3] will contain the actual
1442 1533 4 number of characters returned as part of the escape sequence. Now, in some
1443 1534 4 instances (VT1XX and VT2XX families of terminals) the sequence of interest
1444 1535 4 ends with a semicolon, followed by all manner of other stuff that should be
1445 1536 4 meaningful, but isn't. So the search stops at the semicolon. But, VT52's
1446 1537 4 and other terminals end in a different way, with no semicolon, and/or no
1447 1538 4 other trailing garbage. In those cases, the total length of good stuff
1448 1539 4 is the length found in IOSB[4]. Also, some sequences end with 'c' and no
1449 1540 4 semicolon. Change that 'c' to a ';'. Finally (for now), in some cases when
1450 1541 4 a user is typing while the QIO is performed, garbage comes in before the
1451 1542 4 valid escape sequence. IOSB[1] is the offset into the response buffer that
1452 1543 4 will get us to the beginning of the escape sequence, and IOSB[3] tells how
1453 1544 4 long the escape sequence is.
1454 1545 4
1455 1546 4
1456 1547 4 So, first get to where the "real" response sequence starts
1457 1548 4
1458 1549 4 sequence = resp_buffer[.iosb[1]+1];
1459 1550 4
1460 1551 4 Look for a semicolon. If no semicolon, the IOSB[3] is the length of
1461 1552 4 the string. Make one further check, changing the 'c' at the end (if there
1462 1553 4 is one) to a ';'.
1463 1554 4
1464 1555 4 ptr = CH$FIND (CH(.iosb[3] - 1, .sequence, ';');
1465 1556 4 IF CH$FAIL(.ptr)
1466 1557 4 THEN
1467 1558 5 BEGIN
1468 1559 5 resp_len = .iosb[3] - 1;
1469 1560 5 IF .sequence[resp_len - 1] EQL 'c'
1470 1561 5 THEN sequence[resp_len - 1] = ';';
1471 1562 5 END
1472 1563 5
1473 1564 5 If the semicolon is found, calculate the response string length.
1474 1565 5
```

```
1475      ELSE resp_len = .ptr - .sequence + 1;
1476
1477
1478      Now loop thru all the terminal types, to see if any of them have the same
1479      response string as what we found.
1480
1481      INCR i FROM 0 TO (term$_num - 1) DO          ! Go thru each term block
1482      BEGIN
1483      IF .term$_table[.i, term$_rspnum] NEQ 0      ! If a response block exists
1484      THEN                                         ! then go thru all the responses
1485      BEGIN
1486      BIND resp_block = .term$_table[.i, term$_rspblk] : VECTOR;
1487      INCR j FROM 0 TO (.term$_table[.i, term$_rspnum] - 1) DO
1488      BEGIN
1489      BIND response = .resp_block[.j] : VECTOR;
1490      IF CH$EQL(.response[0], .response[1],
1491      .resp_len, .sequence)
1492      THEN (index = .i; EXITLOOP);
1493      END;
1494      IF .index NEQ -1
1495      THEN EXITLOOP;
1496      END;
1497      END;
1498
1499      If still no match, then return saying that the terminal type is unknown.
1500
1501      IF .index NEQ -1
1502      THEN
1503      BEGIN
1504
1505      A VT100J looks almost like a VK100, except that the character after the
1506      ":", is a 2. If that's what we have, then set it up that way.
1507
1508      IF .index EQL term$_vk100
1509      AND .sequence[.resp_len] EQL '2'
1510      THEN index = term$_vt100;
1511
1512      In the VT1xx family of terminals, the presence of the advanced video
1513      is based on the character AFTER the ":". If this character, translated
1514      to a "real" number, has bit 1 set (i.e. 2,3,6,7) then the terminal has
1515      the advanced video option.
1516
1517      IF .index EQL term$_vt100
1518      OR .index EQL term$_vt101
1519      OR .index EQL term$_vt102
1520      OR .index EQL term$_vt105
1521      OR .index EQL term$_vt125
1522      OR .index EQL term$_vt131
1523      OR .index EQL term$_vt132
1524      OR .index EQL term$_vt173
1525      AND
1526      (
1527      BEGIN
1528      LOCAL char : BYTE;
1529      char = .sequence[.resp_len] - '0';
1530
1531
```



```
: 1532      1623  7
: 1533      1624  7
: 1534      1625  6
: 1535      1626  7
: 1536      1627  7
: 1537      1628  7
: 1538      1629  7
: 1539      1630  7
: 1540      1631  7
: 1541      1632  6
: 1542      1633  5
: 1543      1634  5
: 1544      1635  5
: 1545      1636  5
: 1546      1637  6
: 1547      1638  6
: 1548      1639  6
: 1549      1640  6
: 1550      1641  6
: 1551      1642  6
: 1552      1643  6
: 1553      1644  6
: 1554      1645  6
: 1555      1646  6
: 1556      1647  6
: 1557      1648  6
: 1558      1649  6
: 1559      1650  7
: 1560      1651  7
: 1561      1652  7
: 1562      1653  7
: 1563      1654  7
: 1564      1655  7
: 1565      1656  7
: 1566      1657  7
: 1567      1658  6
: 1568      1659  6
: 1569      1660  5
: 1570      1661  5
: 1571      1662  5
: 1572      1663  5
: 1573      1664  5
: 1574      1665  5
: 1575      1666  5
: 1576      1667  6
: 1577      1668  5
: 1578      1669  6
: 1579      1670  6
: 1580      1671  6
: 1581      1672  6
: 1582      1673  6
: 1583      1674  6
: 1584      1675  6
: 1585      1676  5
: 1586      1677  5
: 1587      1678  5
: 1588      1679  5

      .char<1,1>
      END
OR
      BEGIN
      LOCAL CHAR : VECTOR[3,BYTE];
      CHAR = '.11';
      PTR = CH$FIND_SUB(.IOSB[3]-1, .SEQUENCE, 3, CHAR);
      CH$FAIL(.PTR)
      END
)
THEN tt2_set = .tt2_set OR tt2$m_avo
ELSE tt2_clr = .tt2_clr OR tt2$m_avo;
IF .index EQL term$_VT200_Series
THEN
      BEGIN
      LOCAL x,
      opt200 : VECTOR[8, LONG]
      PRESET ([0] = 0,
              [1] = tt2$m_printer,
              [2] = tt2$m_regis,
              [3] = tt2$m_sixel,
              [4] = 0,
              [5] = 0,
              [6] = tt2$m_drchs,
              [7] = 0);
      tt2_set = .tt2_set OR tt2$m_avo;
      INCR x from 1 TO 8 DO
      BEGIN
      local str: vector[2,byte];
      str[0] = '.';
      str[1] = '0' + .x;
      ptr = CH$FIND_SUB(.iosb[3] - 1, .sequence, 2, str);
      IF NOT CH$FAIL(.ptr)
      THEN
          tt2_set = .tt2_set OR .OPT200[.x - 1];
      END;
      flags[set$v_vt200] = 1;
      END;

      The pro terminal has the terminal characteristic, regis, set if
      the extended bit option in the response string is set ( first byte
      after the actual response sequence). Otherwise the characteristic
      is turned off.

      IF (.index EQL term$_pro_series )
      THEN
      BEGIN
      tt2_set = .tt2_set or tt2$m_avo ;
      IF .sequence[.resp_len] EQL '1'
      THEN
          tt2_set = .tt2_set or tt2$m_regis
      ELSE
          tt2_clr = .tt2_clr or tt2$m_regis ;
      END;

      If the number 3 is present in the response then regis is
      set for the vt102 ( the rainbow terminal can have regis)
```

```
: 1589      1680 5 !
: 1590      1681 5 IF .index EQL term$_vt102
: 1591      1682 5 THEN
: 1592      1683 6 BEGIN
: 1593      1684 6 local str; vector[2,byte] ;
: 1594      1685 6 str = 1,3 ;
: 1595      1686 6 ptr = ch$find_sub( .iosb[3]-1, .sequence, 2, str ) ;
: 1596      1687 6 IF Not ch$fail(.ptr)
: 1597      1688 6 THEN
: 1598      1689 6 tt2_set = .tt2_set or tt2$m_regis ;
: 1599      1690 6 END;
: 1600      1691 5 return 1;
: 1601      1692 4 END;
: 1602      1693 3 END;
: 1603      1694 2 END;
: 1604      1695 2
: 1605      1696 2 If we went thru all the request sequences and nothing was found, then
: 1606      1697 2 signal saying that the terminal type is unknown, and return.
: 1607      1698 2
: 1608      1699 2 tmpblock = .info_block;
: 1609      1700 2 tmpblock+4 = .(info_block+4);
: 1610      1701 2 $qiow (chan = .chan,
P 1702 2 func = io$_setmode,
P 1703 2 p1 = tmpblock);
: 1613      1704 2 SIGNAL(set$_writeerr AND NOT sts$m_severity OR sts$k_warning,
: 1614      1705 2 1, dev_desc,
: 1615      1706 2 set$_unkterm);
: 1616      1707 2 RETURN 0;
: 1617      1708 1 END;
```

.PSECT \$PLITS\$,NOWRT,NOEXE,2

```
00000000 00000000 00100000 02000000 00400000 00000000# 001CC P.ABY: .LONG 0[20]
00000000 00000000 0021C P.ABZ: .LONG 0, 4194304, 33554432, 1048576, 0, 0, -
00000000 00200000 00234 2097152, 0
```

.PSECT \$CODE\$,NOWRT,2

```
OFFC 00000 INQUIRE_TYPE:
5E FF58 CE 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 1452
56 04 AC D0 00007 MOVAB -168(SP), SP
58 08 A6 9E 0000B MOVL DATA_BUFFER, R6 : 1474
OC AE 28 A6 9E 0000F MOVAB 8(R6), R8
59 34 A6 9E 00014 MOVAB 40(R6), 12(SP)
58 AE 0000' CF 0050 8F 28 00018 MOVAB 52(R6), R9
08 AE 01 CE 00021 MOVAB #80, P.ABY, RESP_BUFFER : 1485
0240 31 00025 MNEGL #1, I : 1496
51 08 AE D0 00028 1$: BRW 29$
50 00000000G0041 D0 0002C MOVL I, R1 : 1498
60 DD 00034 MOVL TERMS_REQBLK[R1], R0 : 1506
04 A0 DD 00036 PUSHL (R0)
7E 04 7D 00039 PUSHL 4(R0)
MOVQ #4, -(SP)
```


	7E	50	8F	9A	0003C	MOVZBL	#80, -(SP)		
		6C	AE	9F	00040	PUSHAB	RESP, BUFFER		
			7E	7C	00043	CLRQ	-(SP)		
		70	AE	9F	00045	PUSHAB	IOSB		
	7E	48F7	8F	3C	00048	MOVZWL	#18679, -(SP)		
	7E	38	A6	3C	0004D	MOVZWL	56(R6), -(SP)		
			7E	D4	00051	CLRL	-(SP)		
00000000G	00		0C	FB	00053	CALLS	#12, SYSSQIOW		
04	AE		50	D0	0005A	MOVL	R0, STATUS		
	09	04	AE	E9	0005E	BLBC	STATUS, 2\$	1507	
04	AE	50	AE	3C	00062	MOVZWL	IOSB, STATUS	1508	
	16	04	AE	E8	00067	BLBS	STATUS, 3\$	1509	
0000022C	8F	04	AE	D1	0006B	CMPL	STATUS, #556	1510	
			0C	13	00073	BEQL	3\$		
	3C	04	AE	D1	00075	CMPL	STATUS, #60	1511	
			06	13	00079	BEQL	3\$		
		04	AE	DD	0007B	PUSHL	STATUS	1516	
		0221	31	0007E	BRW	30\$		1514	
	01	08	AE	D1	00081	CMPL	I, #1	1519	
			2E	12	00085	BNEQ	4\$		
	44	AE	0C	BE	D0	00087	MOVL	@12(SP), TMPBLOCK	1521
48	50	OC	AE	04	C1	0008C	ADDL3	#4, 12(SP), R0	1522
	AE	60	00008000	8F	C9	00091	BISL3	#32768, (R0), TMPBLOCK+4	
				7E	7C	0009A	CLRQ	-(SP)	1525
				7E	7C	0009C	CLRQ	-(SP)	
				7E	D4	0009E	CLRL	-(SP)	
		58	AE	9F	000A0	PUSHAB	TMPBLOCK		
			7E	7C	000A3	CLRQ	-(SP)		
	7E		23	7D	000A5	MOVQ	#35, -(SP)		
	7E	38	A6	3C	000A8	MOVZWL	56(R6), -(SP)		
			7E	D4	000AC	CLRL	-(SP)		
00000000G	00		0C	FB	000AE	CALLS	#12, SYSSQIOW		
0000022C	8F	04	AE	D1	000B5	CMPL	STATUS, #556	1527	
			03	12	000BD	BNEQ	5\$		
		01A6	31	000BF	BRW	29\$			
	50	59	AE	9E	000C2	MOVAB	RESP, BUFFER+1, R0	1549	
	57	52	AE	3C	000C6	MOVZWL	IOSB+2, SEQUENCE		
	57	50	C0	000CA	ADDL2	R0, SEQUENCE			
	10	AE	56	AE	3C	000CD	MOVZWL	IOSB+6, 16(SP)	1555
		10	AE	D7	000D2	DECL	16(SP)		
67	10	AE	3B	3A	000D5	LOCC	#59, 16(SP), (SEQUENCE)		
			02	12	000DA	BNEQ	6\$		
			51	D4	000DC	CLRL	R1		
	6E		51	D0	000DE	MOVL	R1, PTR		
			13	12	000E1	BNEQ	7\$	1556	
	5B	10	AE	D0	000E3	MOVL	16(SP), RESP_LEN	1559	
	63	8F	FF	AB47	91	000E7	CMPB	-1(RES_LEN)[SEQUENCE], #99	1560
				0F	12	000ED	BNEQ	8\$	
	FF	AB47		3B	90	000EF	MOVAB	#59, -1(RES_LEN)[SEQUENCE]	1561
				08	11	000F4	BRB	8\$	1556
51		6E		57	C3	000F6	SUBL3	SEQUENCE, PTR, R1	1566
		5B	01	A1	9E	000FA	MOVAB	1(R1), RESP_LEN	
		54		01	CE	000FE	MNEGL	#1, I	1572
				40	11	00101	BRB	13\$	
50		54		1C	C5	00103	MULL3	#28, I, R0	1574
			00000000G	0040	9F	00107	PUSHAB	TERMS, TABLE+20(R0)	
		5A		9E	D0	0010E	MOVL	@(SP)+, R10	

5B	00	04	B0	14	BE	45	D0	00123	10\$:	MOV	20(SP)[J], R0	1577
			60			2D	00128			CMPC5	(R0), 24(R0), #0, RESP_LEN, (SEQUENCE)	1581
			67				0012E					
			05	12	0012F					BNEQ	11\$	
			54	D0	00131					MOV	1, (R9)	1583
			04	11	00134					BRB	12\$	
E9	FFFFFFF	55	5A	F2	00136	11\$:				AOBLSS	R10, J, 10\$	1578
		8F	69	D1	0013A	12\$:				CMPL	(R9), #-1	1585
B8	FFFFFFF	54	0000000G	8F	F3	00143	13\$:			BNEQ	14\$	
		8F	69	D1	0014B	14\$:				AOBLEQ	#TERMS_NUM-1, 1, 9\$	1572
			03	12	00152					CMPL	(R9), #-1	1593
			0111	31	00154					BNEQ	15\$	
	0000000G	8F	69	D1	00157	15\$:				BRW	29\$	
			0D	12	0015E					CMPL	(R9), #TERMS_VK100	1601
		32	6B47	91	00160					BNEQ	16\$	
			07	12	00164					CMPL	(RESP_LEN)[SEQUENCE], #50	1602
		69	0000000G	8F	D0	00166				BNEQ	16\$	
	0000000G	8F	69	D1	0016D	16\$:				MOV	#TERMS_VT100, (R9)	1603
			65	13	00174					CMPL	(R9), #TERMS_VT100	1611
	0000000G	8F	69	D1	00176					BEQ	18\$	
			5C	13	0017D					CMPL	(R9), #TERMS_VT101	1612
	0000000G	8F	69	D1	0017F					BEQ	18\$	
			53	13	00186					CMPL	(R9), #TERMS_VT102	1613
	0000000G	8F	69	D1	00188					BEQ	18\$	
			4A	13	0018F					CMPL	(R9), #TERMS_VT105	1614
	0000000G	8F	69	D1	00191					BEQ	18\$	
			41	13	00198					CMPL	(R9), #TERMS_VT125	1615
	0000000G	8F	69	D1	0019A					BEQ	18\$	
			38	13	001A1					CMPL	(R9), #TERMS_VT131	1616
	0000000G	8F	69	D1	001A3					BEQ	18\$	
			2F	13	001AA					CMPL	(R9), #TERMS_VT132	1617
	0000000G	8F	69	D1	001AC					BEQ	18\$	
			2C	12	001B3					CMPL	(R9), #TERMS_VT173	1618
		50	6B47	30	83	001B5				BNEQ	19\$	
18	AE	10	50	01	E0	001BA				SUBB3	#48, (RESP_LEN)[SEQUENCE], CHAR	1622
	67		18	00	F0	001BE				BBS	#1, CHAR, 18\$	1623
			AE	03	39	001C8				INSV	#3223867, #0, #24, CHAR	1628
				03	13	001CF				MATCHC	#3, CHAR, 16(SP), (SEQUENCE)	1629
			53	03	D0	001D1				BEQ	17\$	
			6E	53	D7	001D4	17\$:			MOV	#3, R3	
				73	3E	001D6				DECL	R3	
			03	06	12	001D9				MOVAV	-(R3), PTR	
				08	88	001DB	18\$:			BNEQ	19\$	1630
			0F	04	11	001DF				BISB2	#8, 3(R8)	1633
			0000000G	08	88	001E1	19\$:			BRB	20\$	
				69	D1	001E5	20\$:			BISB2	#8, 15(R6)	1634
				35	12	001EC				CMPL	(R9), #TERMS_VT200_SERIES	1635
24	AE	0000'	CF	20	28	001EE				BNEQ	24\$	
		03	A8	08	88	001F5				MOVC3	#32, P.ABZ, OPT200	1647
			54	01	D0	001F9				BISB2	#8, 3(R8)	1648
			1C	3B	90	001FC	21\$:			MOV	#1, X	1649
										MOVB	#59, STR	1652

67	1D	AE	54	30	81	00200	ADDB3	#48, X, STR+1	1653		
	10	AE	1C	AE	02	39	00205	MATCHC	#2, STR, 16(SP), (SEQUENCE)	1654	
			53	03	13	0020C	BEQL	22\$			
			6E	02	D0	0020E	MOVL	#2, R3			
				73	3E	00211	MOVAW	-(R3), PTR			
			68	05	13	00214	BEQL	23\$	1655		
	DD		54	20	AE	44	C8	00216	BISL2	OPT200-4[X], (R8)	1657
		1D	A6	08	F3	0021B	AOBLEQ	#8, X, 21\$		1649	
		00000000G	8F	20	88	0021F	BISB2	#32, 29(R6)		1659	
				69	D1	00223	CMPL	(R9), #TERMS_PRO_SERIES		1667	
		03	A8	14	12	0022A	BNEQ	26\$			
			31	08	88	0022C	BISB2	#8, 3(R8)		1670	
				6B	47	91	00230	CMPB	(RESP_LEN)[SEQUENCE], #49		1671
		03	A8	06	12	00234	BNEQ	25\$			
				02	88	00236	BISB2	#2, 3(R8)		1673	
		0F	A6	04	11	0023A	BRB	26\$			
		00000000G	8F	02	88	0023C	BISB2	#2, 15(R6)		1675	
				69	D1	00240	CMPL	(R9), #TERMS_VT102		1681	
				1B	12	00247	BNEQ	28\$			
		20	AE	8F	B0	00249	MOVW	#13115, STR		1685	
67	10	AE	20	AE	02	39	0024F	MATCHC	#2, STR, 16(SP), (SEQUENCE)	1686	
					03	13	00256	BEQL	27\$		
			53	02	D0	00258	MOVL	#2, R3			
			6E	73	3E	0025B	MOVAW	-(R3), PTR			
				04	13	0025E	BEQL	28\$	1687		
		03	A8	02	88	00260	BISB2	#2, 3(R8)		1689	
			50	01	D0	00264	MOVL	#1, R0		1691	
					04	00267	RET				
FDB5	08	AE	01	8F	F1	00268	ACBL	#TERMS_REQNUM-1, #1, 1, 1\$		1496	
			44	BE	D0	00273	MOVL	@12(SP), TMPBLOCK		1699	
		50	OC	04	C1	00278	ADDL3	#4, 12(SP), R0		1700	
			48	60	D0	0027D	MOVL	(R0), TMPBLOCK+4			
				7E	7C	00281	CLRQ	-(SP)		1703	
				7E	7C	00283	CLRQ	-(SP)			
				7E	D4	00285	CLRL	-(SP)			
				58	AE	9F	00287	PUSHAB	TMPBLOCK		
			7E	7E	7C	0028A	CLRQ	-(SP)			
			7E	23	7D	0028C	MOVQ	#35, -(SP)			
				38	A6	3C	0028F	MOVZWL	56(R6), -(SP)		
		00000000G	00	7E	D4	00293	CLRL	-(SP)			
				OC	FB	00295	CALLS	#12, SYSSQIOW			
				8F	DD	0029C	PUSHL	#SET\$ UNKTERM		1704	
				20	A6	9F	002A2	PUSHAB	32(R6)		
					01	DD	002A5	PUSHL	#1		
		00000000G	00	8F	DD	002A7	PUSHL	#<SET\$ WRITEERR-8>			
				04	FB	002AD	CALLS	#4, LIB\$SIGNAL			
				50	D4	002B4	CLRL	R0		1708	
					04	002B6	RET				

; Routine Size: 695 bytes, Routine Base: \$CODE\$ + 0A82

```
1619 1709 1 ROUTINE get_term_def (data_buffer) =
1620 1710 2 BEGIN
1621 1711 2
1622 1712 2 ++
1623 1713 2 Functional description
1624 1714 2
1625 1715 2 This routine searches TERMTABLE.EXE for a terminal definition.
1626 1716 2 We look in TERMTABLE only if we don't recognize the terminal
1627 1717 2 name.
1628 1718 2
1629 1719 2 This routine will extract various information about the terminal
1630 1720 2 from its definition. This information is stored in INFO_BLOCK.
1631 1721 2
1632 1722 2 Inputs
1633 1723 2 DATA_BUFFER - contains all meaningful data
1634 1724 2
1635 1725 2 Outputs
1636 1726 2 fills in various fields in INFO_BLOCK
1637 1727 2 --
1638 1728 2
1639 1729 2 MAP
1640 1730 2 data_buffer : REF VECTOR;
1641 1731 2
1642 1732 2 MACRO
1643 1733 2
1644 1734 2 $first_item(a,b,c,d) =
1645 1735 2 a
1646 1736 2
1647 1737 2 %,
1648 1738 2
1649 1739 2 $cap_init[capability,action,characteristic,longword_number] =
1650 1740 2
1651 1741 2 %NAME(smg$k_,capability)
1652 1742 2
1653 1743 2 %,
1654 1744 2
1655 1745 2 $set_init[capability,action,characteristic,longword_number] =
1656 1746 2
1657 1747 2 $first_item( %NAME(term$l_set,longword_number) )
1658 1748 2
1659 1749 2 %,
1660 1750 2
1661 1751 2 $val_init[capability,action,characteristic,longword_number] =
1662 1752 2
1663 1753 2 %IF %IDENTICAL(action,set)
1664 1754 2 %THEN 1
1665 1755 2 %ELSE 0
1666 1756 2 %FI
1667 1757 2
1668 1758 2 %,
1669 1759 2
1670 1760 2 $msk_init[capability,action,characteristic,longword_number] =
1671 1761 2
1672 1762 2 %NAME( tt,
1673 1763 2
1674 1764 2 %IF %IDENTICAL(longword_number,2) %THEN 2 %FI,
1675 1765 2
```



```
1676 M 1766 2
1677 M 1767 $m_.characteristic)
1678 M 1768
1679 M 1769 %,
1680 M 1770
1681 M 1771 $bits[] =
1682 M 1772
1683 M 1773 LITERAL
1684 M 1774
1685 M 1775 cap_size = %LENGTH/4;
1686 M 1776
1687 M 1777 OWN
1688 M 1778
1689 M 1779 cap_vector : VECTOR[cap_size] INITIAL( $cap_init(%REMAINING) ),
1690 M 1780 set_vector : VECTOR[cap_size] INITIAL( $set_init(%REMAINING) ),
1691 M 1781 val_vector : VECTOR[cap_size] INITIAL( $val_init(%REMAINING) ),
1692 M 1782 msk_vector : VECTOR[cap_size] INITIAL( $msk_init(%REMAINING) );
1693 M 1783
1694 M 1784 %,
1695 M 1785
1696 M 1786
1697 M 1787
1698 M 1788 We need to call SMG$GET_TERM_DATA several times.
1699 M 1789 The code will look neater if we use a macro.
1700 M 1790
1701 M 1791 $get_term_data(request_code) =
1702 M 1792
1703 M 1793 BEGIN
1704 M 1794 status = smg$get_term_data( term_table_addr,
1705 M 1795 %ref(request_code),
1706 M 1796 %ref(4),
1707 M 1797 ret_length,
1708 M 1798 ret_buffer);
1709 M 1799
1710 M 1800 IF NOT .status
1711 M 1801 THEN
1712 M 1802 BEGIN
1713 M 1803 SIGNAL(set$_writeerr AND NOT sts$m_severity OR sts$k_warning,
1714 M 1804 1, dev_desc, .status);
1715 M 1805 RETURN 0;
1716 M 1806 END;
1717 M 1807
1718 M 1808 %;
1719 M 1809
1720 M 1810
1721 M 1811 Bind data to meaningful names
1722 M 1812
1723 M 1813
1724 M 1814 bind_data;
1725 M 1815
1726 M 1816 LOCAL
1727 M 1817 crfill,
1728 M 1818 lffill,
1729 M 1819 frame,
1730 M 1820 status,
1731 M 1821 term_table_addr,
1732 M 1822 ret_length,
```


SETTERM
V04-000

B 14
16-Sep-1984 01:10:06
14-Sep-1984 12:09:20

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETTERM.B32;1

Page 54
(9)

; 1733

1823 2

ret_buffer;

: 1735 1824 2 : This table describes the correspondence between TERMTABLE
: 1736 1825 2 : capability names and terminal characteristics.
: 1737 1826 2 : For example, the ADVANCED VIDEO capability causes the AVO
: 1738 1827 2 : bit in the 2nd terminal characteristics longword to get set.

: 1739 1828 2				
: 1740 P 1829 2	\$BITS(
: 1741 P 1830 2				
: 1742 P 1831 2	! Capability	! Action	! Characteristic	! Longword #
: 1743 P 1832 2				
: 1744 P 1833 2	ADVANCED_VIDEO,	SET,	AVO,	2,
: 1745 P 1834 2	ANSI CRT,	SET,	ANSICRT,	2,
: 1746 P 1835 2	BLOCK MODE,	SET,	BLOCK,	2,
: 1747 P 1836 2	DEC CRT,	SET,	DECCRT,	2,
: 1748 P 1837 2	EDIT,	SET,	EDIT,	2,
: 1749 P 1838 2	EIGHT BIT,	SET,	EIGHTBIT,	1,
: 1750 P 1839 2	PHYSICAL_FF,	SET,	MECHFORM,	1,
: 1751 P 1840 2	FULLDUP,	CLR,	HALFDUP,	1,
: 1752 P 1841 2	LOWERCASE,	SET,	LOWER,	1,
: 1753 P 1842 2	REGIS,	SET,	REGIS,	2,
: 1754 P 1843 2	SCOPE,	SET,	SCOPE,	1,
: 1755 P 1844 2	SIXEL GRAPHICS,	SET,	SIXEL,	2,
: 1756 P 1845 2	SOFT CHARACTERS,	SET,	DRCS,	2,
: 1757 1846 2	PHYSICAL_TABS,	SET,	MECHTAB,	1);


```
1759 1847 2 |
1760 1848 2 |
1761 1849 2 |
1762 1850 2 |
1763 1851 2 |
1764 1852 2 |
1765 1853 2 |
1766 1854 2 |
1767 1855 2 |
1768 1856 2 |
1769 1857 2 |
1770 1858 2 |
1771 1859 2 |
1772 1860 2 |
1773 1861 2 |
1774 1862 2 |
1775 1863 2 |
1776 1864 2 |
1777 1865 2 |
1778 1866 2 |
1779 1867 2 |
1780 1868 2 |
1781 1869 2 |
1782 1870 2 |
1783 1871 2 |
1784 1872 2 |
1785 1873 2 |
1786 1874 2 |
1787 1875 2 |
1788 1876 2 |
1789 1877 2 |
1790 1878 2 |
1791 1879 2 |
1792 1880 2 |
1793 1881 2 |
1794 1882 2 |
1795 1883 2 |
1796 1884 2 |
1797 1885 2 |
1798 1886 2 |
1799 1887 2 |
1800 1888 2 |
1801 1889 2 |
1802 1890 2 |
1803 1891 2 |
1804 1892 2 |
1805 1893 2 |
1806 1894 2 |
1807 1895 2 |
1808 1896 2 |
1809 1897 2 |
1810 1898 2 |
1811 1899 2 |
1812 1900 2 |
1813 1901 2 |
1814 1902 2 |
1815 1903 2 |

On-the-fly activate the SMG package

LIB$FIND_IMAGE SYMBOL(SD SMGSHR,
    %ascid'SMG$INIT_TERM_TABLE', $SMG$INIT_TERM_TABLE);
LIB$FIND_IMAGE SYMBOL(SD SMGSHR,
    %ascid'SMG$DEL_TERM_TABLE', $SMG$DEL_TERM_TABLE);
LIB$FIND_IMAGE SYMBOL(SD SMGSHR,
    %ascid'SMG$GET_TERM_DATA', $SMG$GET_TERM_DATA);

See if this terminal is defined in TERMTABLE.EXE.

IF NOT (smg$init_term_table(name_desc, term_table_addr))
THEN
    BEGIN
        SIGNAL(set$writeerr AND NOT sts$m_severity OR sts$k_warning,
            1, dev_desc,
            set$invquaval,
            2, name_desc, SD_DEVICE_TYPE);
        RETURN 0;
    END;

Found this terminal name in the definition file.
Now get the 'foreign' terminal number assigned to it.

$set_term_data(smg$k_vms_terminal_number);

Store the terminal number

index = .ret_buffer;
info_block[term$b_type] = .ret_buffer<0,8>;
! use this in log_results
! use this in QIOWs

Next get the width and page size of the screen.

$set_term_data(smg$k_columns);

Save the width. Note that this is optional information in
a terminal definition so we may not have received anything
in our buffer.

IF .ret_length NEQ 0
THEN
    BEGIN
        IF .ret_buffer GTRU 511
        THEN
```



```
: 1816      1904  4      BEGIN
: 1817      1905  4      LOCAL   desc   : VECTOR[2],
: 1818      1906  4          buf    : VECTOR[64,BYTE];
: 1819      1907  4      desc[0]=64;
: 1820      1908  4      desc[1]=buf;
: 1821      1909  4      $fao(%ASCII '!UL',desc,desc,.ret_buffer);
: 1822      1910  4      SIGNAL(set$_invquaval, 2, desc, SD_WIDTH);
: 1823      1911  4      RETURN 0
: 1824      1912  4      END;
: 1825      1913  3      info_block[term$w_width] = .ret_buffer<0,16>;
: 1826      1914  2      END;
: 1827      1915  2
: 1828      1916  2      $get_term_data(smg$k_rows);
: 1829      1917  2
: 1830      1918  2      !
: 1831      1919  2      ! Save the page size.
: 1832      1920  2      !
: 1833      1921  2
: 1834      1922  2      IF .ret_length NEQ 0
: 1835      1923  2      THEN
: 1836      1924  3      BEGIN
: 1837      1925  3      IF .ret_buffer GTRU 255
: 1838      1926  3      THEN
: 1839      1927  4      BEGIN
: 1840      1928  4      LOCAL   desc   : VECTOR[2],
: 1841      1929  4          buf    : VECTOR[64,BYTE];
: 1842      1930  4      desc[0]=64;
: 1843      1931  4      desc[1]=buf;
: 1844      1932  4      $fao(%ASCII '!UL',desc,desc,.ret_buffer);
: 1845      1933  4      SIGNAL(set$_invquaval, 2, desc, SD_PAGE);
: 1846      1934  4      RETURN 0
: 1847      1935  3      END;
: 1848      1936  3      info_block[term$b_page] = .ret_buffer <0,8>;
: 1849      1937  2      END;
: 1850      1938  2
: 1851      1939  2      !
: 1852      1940  2      ! Get the fill characteristics.
: 1853      1941  2      !
: 1854      1942  2
: 1855      1943  2      $get_term_data(smg$k_cr_fill);
: 1856      1944  2
: 1857      1945  2      !
: 1858      1946  2      ! Save the carriage return fill count.
: 1859      1947  2      !
: 1860      1948  2
: 1861      1949  2      IF .ret_length EQL 0
: 1862      1950  2      THEN
: 1863      1951  2      crfill=0
: 1864      1952  2      ELSE
: 1865      1953  3      BEGIN
: 1866      1954  3      BIND set1 = info_block[term$l_set1] : $bblock;
: 1867      1955  3      IF .ret_buffer GTRU 9
: 1868      1956  3      THEN
: 1869      1957  4      BEGIN
: 1870      1958  4      LOCAL   desc   : VECTOR[2],
: 1871      1959  4          buf    : VECTOR[64,BYTE];
: 1872      1960  4      desc[0]=64;
```

```
: 1873      1961  4      desc[1]=buf;
: 1874      1962  4      $fao(%ASCII '!UL',desc,desc,.ret_buffer);
: 1875      1963  4      SIGNAL(set$_invquaval, 2, desc, SD_CRFILL);
: 1876      1964  4      RETURN 0
: 1877      1965  3      END;
: 1878      1966  3      crfill = .ret_buffer;
: 1879      1967  3      set1[tt$$_crfill]=1
: 1880      1968  2      END;
: 1881      1969  2
: 1882      1970  2      $get_term_data(smg$_lf_fill);
: 1883      1971  2
: 1884      1972  2      :
: 1885      1973  2      : Save the line feed fill count.
: 1886      1974  2      :
: 1887      1975  2
: 1888      1976  2      IF .ret_length EQL 0
: 1889      1977  2      THEN
: 1890      1978  2          lffill=0
: 1891      1979  2      ELSE
: 1892      1980  3      BEGIN
: 1893      1981  3      BIND set1 = info_block[term$_set1] : $bblock;
: 1894      1982  3      IF .ret_buffer GTRU 9
: 1895      1983  3      THEN
: 1896      1984  4      BEGIN
: 1897      1985  4          LOCAL desc : VECTOR[2],
: 1898      1986  4          buf : VECTOR[64,BYTE];
: 1899      1987  4          desc[0]=64;
: 1900      1988  4          desc[1]=buf;
: 1901      1989  4          $fao(%ASCII '!UL',desc,desc,.ret_buffer);
: 1902      1990  4          SIGNAL(set$_invquaval, 2, desc, SD_LFFILL);
: 1903      1991  4          RETURN 0
: 1904      1992  3      END;
: 1905      1993  3      lffill = .ret_buffer;
: 1906      1994  3      set1[tt$$_lffill]=1
: 1907      1995  2      END;
: 1908      1996  2
: 1909      1997  2      :
: 1910      1998  2      : Combine the resultant fill into one longword.
: 1911      1999  2      :
: 1912      2000  2
: 1913      2001  2      fill = .lffill^8 OR .crfill;
: 1914      2002  2
: 1915      2003  2      :
: 1916      2004  2      : Get the frame count.
: 1917      2005  2      :
: 1918      2006  2
: 1919      2007  2      $get_term_data(smg$_frame);
: 1920      2008  2
: 1921      2009  2      :
: 1922      2010  2      : Save the frame count.
: 1923      2011  2      :
: 1924      2012  2
: 1925      2013  2      IF .ret_length EQL 0
: 1926      2014  2      THEN
: 1927      2015  2          frame=0
: 1928      2016  2      ELSE
: 1929      2017  3      BEGIN
```



```
: 1930      2018  3      frame = .ret_buffer;
: 1931      2019  4      IF NOT (.frame EQL 0 OR (.frame GEQ 5 AND .frame LEQ 8))
: 1932      2020  3      THEN
: 1933      2021  4          BEGIN
: 1934      2022  4              LOCAL desc : VECTOR[2],
: 1935      2023  4                  buf   : VECTOR[64,BYTE];
: 1936      2024  4              desc[0]=64;
: 1937      2025  4              desc[1]=buf;
: 1938      2026  4              $fao(%ASCID '!UL',desc,desc,.frame);
: 1939      2027  4              SIGNAL(set$_invquaval, 2, desc, SD_FRAME);
: 1940      2028  4              RETURN 0
: 1941      2029  3          END;
: 1942      2030  3      parity = tt$m_altframe or parity;
: 1943      2031  3      parity<0,4> = .frame
: 1944      2032  3      END;
: 1945      2033  3
: 1946      2034  3      :
: 1947      2035  3      : Set the terminal characteristics bits that correspond to the various
: 1948      2036  3      : boolean capabilities.
: 1949      2037  3      :
: 1950      2038  3
: 1951      2039  3      INCR index FROM 0 TO cap_size-1 DO
: 1952      2040  3          BEGIN
: 1953      2041  3              LOCAL
: 1954      2042  3                  set_position;
: 1955      2043  3
: 1956      2044  3                  set_position;
: 1957      2045  3
: 1958      2046  3                  $get_term_data(.cap_vector[index]);
: 1959      2047  3
: 1960      2048  3                  set_position = .set_vector[index];
: 1961      2049  3
: 1962      2050  3                  :
: 1963      2051  3                  : If the capability is not defined, then do nothing.
: 1964      2052  3                  : If the capability is defined to be a 1, then set the
: 1965      2053  3                  : bit representing this capability in the appropriate
: 1966      2054  3                  : "set" longword. (This could be term$_set1 or term$_set2.
: 1967      2055  3                  : If the capability is defined to be a 0, then clear the
: 1968      2056  3                  : bit representing this capability in the appropriate "set" longword.
: 1969      2057  3                  : The above action reverses if the val_vector is 0.
: 1970      2058  3                  :
: 1971      2059  3
: 1972      2060  3      IF .ret_length NEQ 0
: 1973      2061  3      THEN
: 1974      2062  4          BEGIN
: 1975      2063  4              BIND
: 1976      2064  4
: 1977      2065  4                  set_longw = info_block[.set_position,0,32,0];
: 1978      2066  4
: 1979      2067  4              IF .ret_buffer EQV .val_vector[index]
: 1980      2068  4              THEN set_longw = .set_longw OR .msk_vector[index]
: 1981      2069  4              ELSE set_longw = .set_longw AND NOT .msk_vector[index]
: 1982      2070  4
: 1983      2071  4              END;
: 1984      2072  3
: 1985      2073  3      END;
: 1986      2074  2      END;
```



```
: 1987      2075 2
: 1988      2076 2
: 1989      2077 2
: 1990      2078 2
: 1991      2079 2
: 1992      2080 2
: 1993      2081 2
: 1994      2082 2
: 1995      2083 2
: 1996      2084 1

! Done with this terminal definition. Get
! rid of the virtual memory used to hold TERMTABLE.EXE.

smg$del_term_table();      ! don't care about return status

RETURN 1;
END;                        ! end of routine get_term_def
```

```
.PSECT $SPLITS,NOWRT,NOEXE,2

54 5F 4D 52 45 54 5F 54 49 4E 49 24 47 4D 53 0023C P.ACB: .ASCII \SMG$INIT_TERM_TABLE\<0>
00 45 4C 42 41 0024B
010E0013 00250 P.ACA: .LONG 17694739
00000000' 00254 .ADDRESS P.ACB
41 54 5F 4D 52 45 54 5F 4C 45 44 24 47 4D 53 00258 P.ACD: .ASCII \SMG$DEL_TERM_TABLE\<0><0>
00 00 45 4C 42 00267
010E0012 0026C P.ACC: .LONG 17694738
00000000' 00270 .ADDRESS P.ACD
41 44 5F 4D 52 45 54 5F 54 45 47 24 47 4D 53 00274 P.ACF: .ASCII \SMG$GET_TERM_DATA\<0><0><0>
00 00 00 41 54 00283
010E0011 00288 P.ACE: .LONG 17694737
00000000' 0028C .ADDRESS P.ACF
00 4C 55 21 00290 P.ACH: .ASCII \!UL\<0>
010E0003 00294 P.ACG: .LONG 17694723
00000000' 00298 .ADDRESS P.ACH
00 4C 55 21 0029C P.ACJ: .ASCII \!UL\<0>
010E0003 002A0 P.ACI: .LONG 17694723
00000000' 002A4 .ADDRESS P.ACJ
00 4C 55 21 002A8 P.ACL: .ASCII \!UL\<0>
010E0003 002AC P.ACK: .LONG 17694723
00000000' 002B0 .ADDRESS P.ACL
00 4C 55 21 002B4 P.ACN: .ASCII \!UL\<0>
010E0003 002B8 P.ACM: .LONG 17694723
00000000' 002BC .ADDRESS P.ACN
00 4C 55 21 002C0 P.ACP: .ASCII \!UL\<0>
010E0003 002C4 P.ACO: .LONG 17694723
00000000' 002C8 .ADDRESS P.ACP
```

```
.PSECT $OWNS,NOEXE,2

00000008 00000007 00000006 00000005 00000002 00000001 0000C CAP_VECTOR:
00000013 00000012 00000011 0000000C 00000009 00000016 00024 .LONG 1, 2, 5, 6, 7, 8, 22, 9, 12, 17, 18, 19, -
00000015 00000014 0003C 20, 21
00000004 00000008 00000008 00000008 00000008 00000008 00044 SET_VECTOR:
00000008 00000004 00000008 00000004 00000004 00000004 0005C .LONG 8, 8, 8, 8, 8, 4, 4, 4, 4, 8, 4, 8, 8, 4
00000001 00000001 00000001 00000001 00000001 00000008 00074
00000001 00000001 00000001 00000001 00000001 00000001 0007C VAL_VECTOR:
00000001 00000001 00000001 00000001 00000000 00000001 00094 .LONG 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1
00000001 00000001 00000001 00000001 00000000 00000001 000AC
```


SETTERM
V04-000

I 14
16-Sep-1984 01:10:06
14-Sep-1984 12:09:20

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETTERM.B32;1

Page 61
(11)

00008000 10000000 20000000 04000000 01000000 08000000 000B4 MSK_VECTOR:
00100000 00001000 02000000 00000080 00100000 00080000 000CC
00000100 00200000 000E4

.LONG 134217728, 16777216, 67108864, 536870912, - ;
268435456, 32768, 524288, 1048576, 128, - ;
33554432, 4096, 1048576, 2097152, 256 ;

.EXTRN SYSSFAO

.PSECT \$CODE\$,NOWRT,2

OFFC 00000 GET_TERM_DEF:

5B	00000000G	00	9E	00002	WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	1709
5A	00000000G	00	9E	00009	MOVAB	LIB\$FIND_IMAGE_SYMBOL, R11	
59	0000	CF	9E	00010	MOVAB	SYSSFAO, R10	
58	0000	CF	9E	00015	MOVAB	\$SMG\$GET_TERM_DATA, R9	
5E	A4	AE	9E	0001A	MOVAB	SD_SMGSHR, R8	
54	04	AC	DO	0001E	MOVL	-92(SP), SP	
56	20	A4	9E	00022	MOVAB	DATA_BUFFER, R4	1808
53	28	A4	9E	00026	MOVAB	32(R4), R6	
	F8	A9	9F	0002A	MOVAB	40(R4), R3	
	0248	C8	9F	0002D	PUSHAB	\$SMG\$INIT_TERM_TABLE	1851
		58	DD	00031	PUSHAB	P.ACA	
6B		03	FB	00033	PUSHL	R8	
	FC	A9	9F	00036	CALLS	#3, LIB\$FIND_IMAGE_SYMBOL	
	0264	C8	9F	00039	PUSHAB	\$SMG\$DEL_TERM_TABLE	1853
		58	DD	0003D	PUSHAB	P.ACC	
6B		03	FB	0003F	PUSHL	R8	
	0280	59	DD	00042	CALLS	#3, LIB\$FIND_IMAGE_SYMBOL	
		C8	9F	00044	PUSHL	R9	1855
		58	DD	00048	PUSHAB	P.ACE	
6B		03	FB	0004A	PUSHL	R8	
	10	AE	9F	0004D	CALLS	#3, LIB\$FIND_IMAGE_SYMBOL	
	44	A4	9F	00050	PUSHAB	TERM_TABLE_ADDR	1862
F8	B9	02	FB	00053	PUSHAB	68(R4)	
23		50	EB	00057	CALLS	#2, \$SMG\$INIT_TERM_TABLE	
	00AC	C8	9F	0005A	BLBS	R0, 1\$	
	44	A4	9F	0005E	PUSHAB	SD_DEVICE_TYPE	1865
		02	DD	00061	PUSHAB	68(R4)	
	0077132A	8F	DD	00063	PUSHL	#2	
		56	DD	00069	PUSHL	#7803690	
		01	DD	0006B	PUSHL	R6	
	00000000*	8F	DD	0006D	PUSHL	#1	
00000000G	00	07	FB	00073	PUSHL	#<SET\$ WRITEERR8-8>	
		0272	31	0007A	CALLS	#7, LIB\$SIGNAL	
		08	AE	9F	BRW	30\$	1869
		10	AE	9F	PUSHAB	RET_BUFFER	1877
		04	DO	00083	PUSHAB	RET_LENGTH	
OC	AE	0C	AE	9F	MOVL	#4, -12(SP)	
		E3	8F	9A	PUSHAB	12(SP)	
OC	AE	0C	AE	9F	MOVZBL	#227, 12(SP)	
		20	AE	9F	PUSHAB	12(SP)	
00	B9	05	FB	00095	PUSHAB	TERM_TABLE_ADDR	
	55	50	DO	00099	CALLS	#5, \$SMG\$GET_TERM_DATA	
	29	55	E9	0009C	MOVL	R0, STATUS	
34	A4	08	AE	DO	BLBC	STATUS, 2\$	
01	A3	08	AE	90	MOVL	RET_BUFFER, 52(R4)	1883
		08	AE	9F	MOVB	RET_BUFFER, 1(R3)	1884
					PUSHAB	RET_BUFFER	1891

		10	AE	9F	000AC	PUSHAB	RET_LENGTH	
	0C		AE	04	D0	000AF	MOVL	#4, -12(SP)
		0C	AE	9F	000B3	PUSHAB	12(SP)	
	0C		AE	8F	9A	000B6	MOVZBL	#221, 12(SP)
		0C	AE	9F	000BB	PUSHAB	12(SP)	
		20	AE	9F	000BE	PUSHAB	TERM_TABLE_ADDR	
	00		B9	05	FB	000C1	CALLS	#5, \$\$\$MSG\$GET_TERM_DATA
			55	50	D0	000C5	MOVL	R0, STATUS
			5E	55	E9	000C8	BLBC	STATUS, 6\$
						2\$:		
		0C	AE	D5	000CB	TSTL	RET_LENGTH	1899
			3A	13	000CE	BEQL	5\$	
000001FF		08	AE	D1	000D0	CMPL	RET_BUFFER, #511	1902
			2B	1B	000D8	BLEQU	4\$	
54	AE	40	8F	9A	000DA	MOVZBL	#64, DESC	1907
58	AE	14	AE	9E	000DF	MOVAB	BUF, DESC+4	1908
		08	AE	DD	000E4	PUSHL	RET_BUFFER	1909
		58	AE	9F	000E7	PUSHAB	DESC	
		5C	AE	9F	000EA	PUSHAB	DESC	
		028C	C8	9F	000ED	PUSHAB	P.ACG	
	6A		04	FB	000F1	CALLS	#4, SYSS\$FAO	
		4C	A8	9F	000F4	PUSHAB	SD_WIDTH	1910
		58	AE	9F	000F7	PUSHAB	DESC	
						3\$:		
			02	DD	000FA	PUSHL	#2	
		0077132A	8F	DD	000FC	PUSHL	#7803690	
			01AD	31	00102	BRW	26\$	
						4\$:		
02	A3	08	AE	B0	00105	MOVW	RET_BUFFER, 2(R3)	1913
		08	AE	9F	0010A	PUSHAB	RET_BUFFER	1916
		10	AE	9F	0010D	PUSHAB	RET_LENGTH	
						5\$:		
0C	AE		04	D0	00110	MOVL	#4, -12(SP)	
		0C	AE	9F	00114	PUSHAB	12(SP)	
0C	AE	E2	8F	9A	00117	MOVZBL	#226, 12(SP)	
		0C	AE	9F	0011C	PUSHAB	12(SP)	
		20	AE	9F	0011F	PUSHAB	TERM_TABLE_ADDR	
	00		B9	05	FB	00122	CALLS	#5, \$\$\$MSG\$GET_TERM_DATA
			55	50	D0	00126	MOVL	R0, STATUS
			52	55	E9	00129	BLBC	STATUS, 9\$
						6\$:		
		0C	AE	D5	0012C	TSTL	RET_LENGTH	1922
			2E	13	0012F	BEQL	8\$	
000000FF		08	AE	D1	00131	CMPL	RET_BUFFER, #255	1925
			1F	1B	00139	BLEQU	7\$	
54	AE	40	8F	9A	0013B	MOVZBL	#64, DESC	1930
58	AE	14	AE	9E	00140	MOVAB	BUF, DESC+4	1931
		08	AE	DD	00145	PUSHL	RET_BUFFER	1932
		58	AE	9F	00148	PUSHAB	DESC	
		5C	AE	9F	0014B	PUSHAB	DESC	
		0298	C8	9F	0014E	PUSHAB	P.ACI	
	6A		04	FB	00152	CALLS	#4, SYSS\$FAO	
		3C	A8	9F	00155	PUSHAB	SD_PAGE	1933
			9D	11	00158	BRB	3\$	
						7\$:		
07	A3	08	AE	90	0015A	MOVW	RET_BUFFER, 7(R3)	1936
		08	AE	9F	0015F	PUSHAB	RET_BUFFER	1943
		10	AE	9F	00162	PUSHAB	RET_LENGTH	
						8\$:		
0C	AE		04	D0	00165	MOVL	#4, -12(SP)	
		0C	AE	9F	00169	PUSHAB	12(SP)	
0C	AE	DE	8F	9A	0016C	MOVZBL	#222, 12(SP)	
		0C	AE	9F	00171	PUSHAB	12(SP)	
		20	AE	9F	00174	PUSHAB	TERM_TABLE_ADDR	

00	B9	05	FB	00177	CALLS	#5, @SSMSG\$GET_TERM_DATA	:
	55	50	D0	0017B	MOVL	R0, STATUS	:
	55	55	E9	0017E	BLBC	STATUS, 13\$:
		0C	AE	D5	TSTL	RET_LENGTH	1949
			04	12	BNEQ	10\$:
			57	D4	CLRL	CRFILL	1951
			2D	11	BRB	12\$:
	09	08	AE	D1	CMPL	RET_BUFFER, #9	1955
			1F	1B	BLEQU	11\$:
54	AE	40	8F	9A	MOVZBL	#64, DESC	1960
58	AE	14	AE	9E	MOVAB	BUF, DESC+4	1961
		08	AE	DD	PUSHL	RET_BUFFER	1962
		58	AE	9F	PUSHAB	DESC	:
		5C	AE	9F	PUSHAB	DESC	:
		02A4	C8	9F	PUSHAB	P.ACK	:
	6A		04	FB	CALLS	#4, SYSS\$FAO	:
		5C	A8	9F	PUSHAB	SD CRFILL	1963
			56	11	BRB	15\$:
	57	08	AE	D0	MOVL	RET_BUFFER, CRFILL	1966
05	A3		04	88	BISB2	#4, -5(R3)	1967
		08	AE	9F	PUSHAB	RET_BUFFER	1970
		10	AE	9F	PUSHAB	RET_LENGTH	:
0C	AE		04	D0	MOVL	#4, -12(SP)	:
		0C	AE	9F	PUSHAB	12(SP)	:
0C	AE	E0	8F	9A	MOVZBL	#224, 12(SP)	:
		0C	AE	9F	PUSHAB	12(SP)	:
		20	AE	9F	PUSHAB	TERM_TABLE_ADDR	:
00	B9	05	FB	001CF	CALLS	#5, @SSMSG\$GET_TERM_DATA	:
	55	50	D0	001D3	MOVL	R0, STATUS	:
	5E	55	E9	001D6	BLBC	STATUS, 18\$:
		0C	AE	D5	TSTL	RET_LENGTH	1976
			04	12	BNEQ	14\$:
			52	D4	CLRL	LFFILL	1978
			2D	11	BRB	17\$:
	09	08	AE	D1	CMPL	RET_BUFFER, #9	1982
			1F	1B	BLEQU	16\$:
54	AE	40	8F	9A	MOVZBL	#64, DESC	1987
58	AE	14	AE	9E	MOVAB	BUF, DESC+4	1988
		08	AE	DD	PUSHL	RET_BUFFER	1989
		58	AE	9F	PUSHAB	DESC	:
		5C	AE	9F	PUSHAB	DESC	:
		02B0	C8	9F	PUSHAB	P.ACM	:
	6A		04	FB	CALLS	#4, SYSS\$FAO	:
		6C	A8	9F	PUSHAB	SD LFFILL	1990
			68	11	BRB	21\$:
	52	08	AE	D0	MOVL	RET_BUFFER, LFFILL	1993
05	A3		08	88	BISB2	#8, -5(R3)	1994
	52		08	78	ASHL	#8, R2, R2	2001
	52		57	C9	BISL3	CRFILL, R2, 24(R4)	:
		08	AE	9F	PUSHAB	RET_BUFFER	2007
		10	AE	9F	PUSHAB	RET_LENGTH	:
0C	AE		04	D0	MOVL	#4, -12(SP)	:
		0C	AE	9F	PUSHAB	12(SP)	:
0C	AE	DF	8F	9A	MOVZBL	#223, 12(SP)	:
		0C	AE	9F	PUSHAB	12(SP)	:
		20	AE	9F	PUSHAB	TERM_TABLE_ADDR	:
00	B9	05	FB	00230	CALLS	#5, @SSMSG\$GET_TERM_DATA	:

```

L 14
16-Sep-1984 01:10:06      VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:09:20      [CLIUTL.SRC]SETTERM.B32:1

```

Page 64
(11)

			55		50	D0	00234		MOVL	R0, STATUS		
			6C		55	E9	00237	18\$:	BLBC	STATUS, 25\$		
				0C	AE	D5	0023A		TSTL	RET_LENGTH		2013
					04	12	0023D		BNEQ	19\$-		
					52	D4	0023F		CLRL	FRAME		2015
					3E	11	00241		BRB	23\$		
			52	08	AE	D0	00243	19\$:	MOVL	RET_BUFFER, FRAME		2018
					29	13	00247		BEQL	22\$		2019
			05		52	D1	00249		CMPL	FRAME, #5		
					05	19	0024C		BLSS	20\$		
			08		52	D1	0024E		CMPL	FRAME, #8		
					1F	15	00251		BLEQ	22\$		
		54	AE	40	8F	9A	00253	20\$:	MOVZBL	#64, DESC		2024
		58	AE	14	AE	9E	00258		MOVAB	BUF, DESC+4		2025
					52	DD	0025D		PUSHL	FRAME		2026
				58	AE	9F	0025F		PUSHAB	DESC		
				5C	AE	9F	00262		PUSHAB	DESC		
				02BC	C8	9F	00265		PUSHAB	P.ACO		
			6A		04	FB	00269		CALLS	#4, SYSSFAO		
				30	A8	9F	0026C		PUSHAB	SD_FRAME		2027
					FE85	31	0026F	21\$:	BRW	3\$-		
			50	14	A4	9E	00272	22\$:	MOVAB	20(R4), R0		2030
			50		10	C9	00276		BISL3	#16, R0, 20(R4)		
			00		52	F0	0027B		INSV	FRAME, #0, #4, 20(R4)		2031
					52	D4	00281	23\$:	CLRL	INDEX		2039
				08	AE	9F	00283	24\$:	PUSHAB	RET_BUFFER		2046
				10	AE	9F	00286		PUSHAB	RET_LENGTH		
		0C	AE		04	D0	00289		MOVL	#4, -12(SP)		
				0C	AE	9F	0028D		PUSHAB	12(SP)		
					A942	D0	00290		MOVL	CAP_VECTOR[INDEX], 12(SP)		
				0C	AE	9F	00296		PUSHAB	12(SP)		
				20	AE	9F	00299		PUSHAB	TERM_TABLE_ADDR		
		00	B9		05	FB	0029C		CALLS	#5, @SMGSGET_TERM_DATA		
			55		50	D0	002A0		MOVL	R0, STATUS		
			15		55	E8	002A3		BLBS	STATUS, 27\$		
					55	DD	002A6	25\$:	PUSHL	STATUS		
					56	DD	002A8		PUSHL	R6		
					01	DD	002AA		PUSHL	#1		
				000000000*	8F	DD	002AC		PUSHL	#<SET\$ WRITEERR&-8>		
		000000000G	00		04	FB	002B2	26\$:	CALLS	#4, LIB\$SIGNAL		
					34	11	002B9		BRB	30\$		
			50	3C	A942	D0	002BB	27\$:	MOVL	SET_VECTOR[INDEX], SET_POSITION		2048
				0C	AE	D5	002C0		TSTL	RET_LENGTH		2060
					1E	13	002C3		BEQL	29\$		
		51	08	AE	74	A942	C1	002C5	ADDL3	VAL_VECTOR[INDEX], RET_BUFFER, R1		2068
				0B	51	E8	002CC		BLBS	R1, -28\$		
					6043	9F	002CF		PUSHAB	(SET_POSITION)[R3]		2069
			9E	00AC	C942	C8	002D2		BISL2	MSK_VECTOR[INDEX], @ (SP)+		
					09	11	002D8		BRB	29\$		
					6043	9F	002DA	28\$:	PUSHAB	(SET_POSITION)[R3		

SETTERM
V04-000

M 14
16-Sep-1984 01:10:06
14-Sep-1984 12:09:20

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETTERM.B32;1

Page 65
(11)

; Routine Size: 754 bytes, Routine Base: \$CODE\$ + 0D39

```
1998 2085 1 ROUTINE log_results (data_buffer) : NOVALUE =
1999 2086 BEGIN
2000 2087
2001 2088 !++
2002 2089 Functional description
2003 2090
2004 2091 This routine tells the user whatever was set.
2005 2092
2006 2093 Inputs
2007 2094 DATA_BUFFER - full of all sorts of meaningful data
2008 2095
2009 2096 Outputs
2010 2097 None.
2011 2098
2012 2099 -----
2013 2100
2014 2101 MAP
2015 2102 data_buffer : REF VECTOR;
2016 2103
2017 2104 LOCAL
2018 2105 desc : $BBLOCK[dsc$ s_bln],
2019 2106 fao_buffer : VECTOR[8];
2020 2107 fao_desc : VECTOR[2];
2021 2108
2022 2109
2023 2110 Bind data_buffer to meaningful names.
2024 2111
2025 2112 bind_data;
2026 2113
2027 2114
2028 2115 Initialize the descriptors
2029 2116
2030 2117 $init_dyndesc(desc);
2031 2118 fao_desc[1] = fao_buffer;
2032 2119
2033 2120
2034 2121 See if a specific terminal type was set.
2035 2122
2036 2123 IF .index GTR -1
2037 2124 THEN
2038 2125 BEGIN
2039 2126 str$append(desc, .term$ name[.index]);
2040 2127 str$append(desc, SD_COMMA);
2041 2128 END;
2042 2129
2043 2130
2044 2131 If this is an unknown terminal that is defined in
2045 2132 TERMTABLE, use the name from the definition.
2046 2133
2047 2134 IF .index LSS -1
2048 2135 THEN
2049 2136 BEGIN
2050 2137 str$append (desc, name_desc);
2051 2138 str$append (desc, SD_COMMA);
2052 2139 END;
2053 2140
2054 2141 !
```



```
2055 2142 2 ! Go thru all 4 of the terminal flagwords, to produce a string showing
2056 2143 2 ! everything that was changed.
2057 2144 2
2058 2145 2 INCR i FROM 0 TO term$_ttset_num - 1 DO
2059 2146 2 BEGIN
2060 2147 2 IF (.tt1_set AND .term$_ttset_bit[i]) NEQ 0
2061 2148 2 THEN
2062 2149 2 BEGIN
2063 2150 2 str$append(desc, .term$_ttset_key[i]);
2064 2151 2 str$append(desc, SD_COMMA);
2065 2152 2 END;
2066 2153 2 END;
2067 2154 2
2068 2155 2 INCR i FROM 4 TO term$_ttclr_num - 1 DO
2069 2156 2 BEGIN
2070 2157 2 IF (.tt1_clr AND .term$_ttclr_bit[i]) NEQ 0
2071 2158 2 THEN
2072 2159 2 BEGIN
2073 2160 2 str$append(desc, .term$_ttclr_key[i]);
2074 2161 2 str$append(desc, SD_COMMA);
2075 2162 2 END;
2076 2163 2 END;
2077 2164 2
2078 2165 2 INCR i FROM 0 TO term$_tt2set_num - 1 DO
2079 2166 2 BEGIN
2080 2167 2 IF (.tt2_set AND .term$_tt2set_bit[i]) NEQ 0
2081 2168 2 THEN
2082 2169 2 BEGIN
2083 2170 2 str$append(desc, .term$_tt2set_key[i]);
2084 2171 2 str$append(desc, SD_COMMA);
2085 2172 2 END;
2086 2173 2 END;
2087 2174 2
2088 2175 2 INCR i FROM 2 TO term$_tt2clr_num - 1 DO
2089 2176 2 BEGIN
2090 2177 2 IF (.tt2_clr AND .term$_tt2clr_bit[i]) NEQ 0
2091 2178 2 THEN
2092 2179 2 BEGIN
2093 2180 2 str$append(desc, .term$_tt2clr_key[i]);
2094 2181 2 str$append(desc, SD_COMMA);
2095 2182 2 END;
2096 2183 2 END;
2097 2184 2
2098 2185 2 INCR i FROM 0 TO term$_ttset_num - 1 DO
2099 2186 2 BEGIN
2100 2187 2 IF (.tt1_clr AND .term$_ttset_bit[i]) NEQ 0
2101 2188 2 THEN
2102 2189 2 BEGIN
2103 2190 2 str$append(desc, SD_NO);
2104 2191 2 str$append(desc, .term$_ttset_key[i]);
2105 2192 2 str$append(desc, SD_COMMA);
2106 2193 2 END;
2107 2194 2 END;
2108 2195 2
2109 2196 2 INCR i FROM 4 TO term$_ttclr_num - 1 DO
2110 2197 2 BEGIN
2111 2198 2 IF (.tt1_set AND .term$_ttclr_bit[i]) NEQ 0
```



```
2112 2199 3 THEN
2113 2200 4 BEGIN
2114 2201 4 str$append(desc, SD_NO);
2115 2202 4 str$append(desc, .term$ ttclr_key[i]);
2116 2203 4 str$append(desc, SD_COMMA);
2117 2204 4 END;
2118 2205 3 END;
2119 2206 3
2120 2207 3 INCR i FROM 0 TO term$ tt2set_num - 1 DO
2121 2208 4 BEGIN
2122 2209 4 IF (.tt2_clr AND .term$ tt2set_bit[i]) NEQ 0
2123 2210 4 THEN
2124 2211 4 BEGIN
2125 2212 4 str$append(desc, SD_NO);
2126 2213 4 str$append(desc, .term$ tt2set_key[i]);
2127 2214 4 str$append(desc, SD_COMMA);
2128 2215 4 END;
2129 2216 3 END;
2130 2217 3
2131 2218 3 INCR i FROM 2 TO term$ tt2clr_num - 1 DO
2132 2219 4 BEGIN
2133 2220 4 IF (.tt2_set AND .term$ tt2clr_bit[i]) NEQ 0
2134 2221 4 THEN
2135 2222 4 BEGIN
2136 2223 4 str$append(desc, SD_NO);
2137 2224 4 str$append(desc, .term$ tt2clr_key[i]);
2138 2225 4 str$append(desc, SD_COMMA);
2139 2226 4 END;
2140 2227 3 END;
2141 2228 3
2142 2229 3
2143 2230 3 Check the special parameters, the ones that take a parameter.
2144 2231 3
2145 2232 3 Dec_Crt
2146 2233 3
2147 2234 3 IF ( tt2$m_deccrt AND .deccrt_set) NEQ 0 THEN !/DEC_CRT
2148 2235 4 BEGIN
2149 2236 4 str$append( desc, SD_DEC CRT ) ;
2150 2237 4 str$append( desc, SD_COMMA ) ;
2151 2238 4 END;
2152 2239 3
2153 2240 3 IF ( tt2$m_deccrt2 AND .deccrt_set) NEQ 0 THEN !/DEC_CRT2
2154 2241 4 BEGIN
2155 2242 4 str$append( desc, SD_DEC CRT2 ) ;
2156 2243 4 str$append( desc, SD_COMMA ) ;
2157 2244 4 END;
2158 2245 3
2159 2246 3 IF (tt2$m_deccrt AND .deccrt_clr) NEQ 0 THEN !/NODEC_CRT
2160 2247 4 BEGIN
2161 2248 4 str$append( desc, SD_NO ) ;
2162 2249 4 str$append( desc, SD_DEC CRT ) ;
2163 2250 4 str$append( desc, SD_COMMA ) ;
2164 2251 4 END;
2165 2252 3
2166 2253 3 IF (tt2$m_deccrt2 AND .deccrt_clr) NEQ 0 THEN !/NODEC_CRT2
2167 2254 4 BEGIN
2168 2255 4 str$append( desc, SD_NO ) ;
```



```
2169      2256      str$append( desc, SD_DEC CRT2 ) ;
2170      2257      str$append( desc, SD_COMMA ) ;
2171      2258      END;
2172      2259      ---
2173      2260      Parity
2174      2261      ---
2175      2262      IF .flags[set$V_nopar]
2176      2263      THEN
2177      2264      BEGIN
2178      2265      str$append(desc, %ASCID 'NOPARITY');
2179      2266      str$append(desc, SD_COMMA);
2180      2267      END
2181      2268      ELSE IF .flags[set$V_odd] OR .flags[set$V_even]
2182      2269      THEN
2183      2270      BEGIN
2184      2271      str$append(desc, SD_PARITY);
2185      2272      str$append(desc, %ASCID '=');
2186      2273      IF .flags[set$V_odd]
2187      2274      THEN str$append(desc, term$_odd)
2188      2275      ELSE str$append(desc, term$_even);
2189      2276      str$append(desc, SD_COMMA);
2190      2277      END;
2191      2278      ---
2192      2279      CRfill
2193      2280      ---
2194      2281      IF .flags[set$V_cr]
2195      2282      THEN
2196      2283      BEGIN
2197      2284      fao_desc[0] = %ALLOCATION(fao_buffer);
2198      2285      $FAOL(CTRSTR = %ASCID 'CRFILL=UL',
2199      2286      $FAOL(CTRSTR = %ASCID 'CRFILL=UL',
2200      2287      OUTBUF = fao_desc,
2201      2288      OUTLEN = fao_desc,
2202      2289      PRMLST = %REF(.fill<0,8>));
2203      2290      str$append(desc, fao_desc);
2204      2291      END;
2205      2292      ---
2206      2293      frame size
2207      2294      ---
2208      2295      IF .flags[set$V_frame]
2209      2296      THEN
2210      2297      BEGIN
2211      2298      fao_desc[0] = %ALLOCATION(fao_buffer);
2212      2299      $FAOL(CTRSTR = %ASCID 'FRAME=UL',
2213      2300      $FAOL(CTRSTR = %ASCID 'FRAME=UL',
2214      2301      OUTBUF = fao_desc,
2215      2302      OUTLEN = fao_desc,
2216      2303      PRMLST = %REF(.PARITY<0,4>));
2217      2304      str$append(desc, fao_desc);
2218      2305      END;
2219      2306      ---
2220      2307      dismiss parity errors
2221      2308      IF .flags[set$V_dismis]
2222      2309      THEN
2223      2310      BEGIN
2224      2311      str$append(desc, %ASCID 'DISMISS PARITY ERRORS');
2225      2312      str$append(desc, SD_COMMA);
```



```
2226 2313 3      END
2227 2314 2  ELSE IF .flags[set$v_nodism]
2228 2315 2  THEN
2229 2316 2  BEGIN
2230 2317 2  str$append(desc, %ASCID 'NO DISMISS PARITY ERRORS');
2231 2318 2  str$append(desc, SD_COMMA);
2232 2319 2  END;
2233 2320 2  --
2234 2321 2  Lf fill
2235 2322 2  --
2236 2323 2  IF .flags[set$v_lf]
2237 2324 2  THEN
2238 2325 2  BEGIN
2239 2326 2  fao_desc[0] = %ALLOCATION(fao_buffer);
2240 2327 2  $FAOL(CTRSTR = %ASCID 'LFFILL=!UL,',
2241 2328 2  OUTBUF = fao_desc,
2242 2329 2  OUTLEN = fao_desc,
2243 2330 2  PRMLST = %REF(.fill<8,8>));
2244 2331 2  str$append(desc, fao_desc);
2245 2332 2  END;
2246 2333 2  --
2247 2334 2  Page size
2248 2335 2  --
2249 2336 2  IF .flags[set$v_page]
2250 2337 2  THEN
2251 2338 2  BEGIN
2252 2339 2  fao_desc[0] = %ALLOCATION(fao_buffer);
2253 2340 2  $FAOL(CTRSTR = %ASCID 'PAGE=!OL,',
2254 2341 2  OUTBUF = fao_desc,
2255 2342 2  OUTLEN = fao_desc,
2256 2343 2  PRMLST = %REF(.info_block[term$b_page]));
2257 2344 2  str$append(desc, fao_desc);
2258 2345 2  END;
2259 2346 2  --
2260 2347 2  Width
2261 2348 2  --
2262 2349 2  IF .flags[set$v_width]
2263 2350 2  THEN
2264 2351 2  BEGIN
2265 2352 2  fao_desc[0] = %ALLOCATION(fao_buffer);
2266 2353 2  $FAOL(CTRSTR = %ASCID 'WIDTH=!UL,',
2267 2354 2  OUTBUF = fao_desc,
2268 2355 2  OUTLEN = fao_desc,
2269 2356 2  PRMLST = %REF(.info_block[term$w_width]));
2270 2357 2  str$append(desc, fao_desc);
2271 2358 2  END;
2272 2359 2  --
2273 2360 2  Speed
2274 2361 2  --
2275 2362 2  IF .flags[set$v_speed]
2276 2363 2  THEN
2277 2364 2  BEGIN
2278 2365 2  LOCAL
2279 2366 2  speeds : VECTOR[2];
2280 2367 2
2281 2368 2
2282 2369 2
```



```
2283      2370      3      fao_desc[0] = %ALLOCATION(fao_buffer);
2284      2371      speeds[0] = .term$.spdblk[.speed<0,8>];
2285      2372      IF .speed<8,8> NEQ 0
2286      2373      THEN speeds[1] = .term$.spdblk[.speed<8,8>]
2287      2374      ELSE speeds[1] = .speeds[0];
2288      2375      $FAOL(CTRSTR = %ASCII 'SPEED=(!AS,!AS)',
2289      2376      OUTBUF = fao_desc,
2290      2377      OUTLEN = fao_desc,
2291      2378      PRMLST = speeds);
2292      2379      str$append(desc, fao_desc);
2293      2380      END;
2294      2381
2295      2382      !
2296      2383      ! Finally, signal an informational message.
2297      2384
2298      2385      IF .desc[dsc$w_length] NEQ 0
2299      2386      THEN
2300      2387      BEGIN
2301      2388      desc[dsc$w_length] = .desc[dsc$w_length] - 1;
2302      2389      SIGNAL(set$_termset, 2, dev_desc, desc);
2303      2390      END;
2304      2391      RETURN;
2305      2392      1 END;
```

.PSECT \$SPLITS,NOWRT,NOEXE,2

59	54	49	52	41	50	4F	4E	002CC	P.ACR:	.ASCII	\NOPARITY\							
						010E0008	002D4	P.ACQ:	.LONG	17694728								
						00000000	002D8		.ADDRESS	P.ACR								
				00	00	00	3D	002DC	P.ACT:	.ASCII	\=<0><0><0>							
						010E0001	002E0	P.ACS:	.LONG	17694721								
						00000000	002E4		.ADDRESS	P.ACT								
00	2C	4C	55	21	3D	4C	4C	49	46	52	43	002E8	P.ACV:	.ASCII	\CRFILL=!UL,\<0>			
						010E000B	002F4	P.ACU:	.LONG	17694731								
						00000000	002F8		.ADDRESS	P.ACV								
00	00	2C	4C	55	21	3D	45	4D	41	52	46	002FC	P.ACX:	.ASCII	\FRAME=!UL,\<0><0>			
						010E000A	00308	P.ACW:	.LONG	17694730								
						00000000	0030C		.ADDRESS	P.ACX								
20	59	54	49	52	41	50	20	53	53	49	4D	53	49	44	00310	P.ACZ:	.ASCII	\DISMISS PARITY ERRORS\<0><0><0>
						00	00	00	53	52	4F	52	52	45	0031F			
												010E0015	00328	P.ACY:	.LONG	17694741		
												00000000	0032C		.ADDRESS	P.ACZ		
49	52	41	50	20	53	53	49	4D	53	49	44	20	4F	4E	00330	P.ADB:	.ASCII	\NO DISMISS PARITY ERRORS\
						53	52	4F	52	52	45	20	59	54	0033F			
												010E0018	00348	P.ADA:	.LONG	17694744		
												00000000	0034C		.ADDRESS	P.ADB		
00	2C	4C	55	21	3D	4C	4C	49	46	46	4C	00350	P.ADD:	.ASCII	\LFFILL=!UL,\<0>			
												010E000B	0035C	P.ADC:	.LONG	17694731		
												00000000	00360		.ADDRESS	P.ADD		
00	00	00	2C	4C	55	21	3D	45	47	41	50	00364	P.ADF:	.ASCII	\PAGE=!UL,\<0><0><0>			
												010E0009	00370	P.ADE:	.LONG	17694729		
												00000000	00374		.ADDRESS	P.ADF		
00	00	2C	4C	55	21	3D	48	54	44	49	57	00378	P.ADH:	.ASCII	\WIDTH=!UL,\<0><0>			
												010E000A	00384	P.ADG:	.LONG	17694730		
												00000000	00388		.ADDRESS	P.ADH		

29 53 41 21 2C 53 41 21 28 3D 44 45 45 50 53 0038C P.ADJ: .ASCII \SPEED=(!AS,!AS),\

010E0010 0039B
00000000 0039C P.ADI: .LONG 17694736
003A0 .ADDRESS P.ADJ

.EXTRN SYSSFAOL

.PSECT \$CODE\$,NOWRT,2

OFFC 00000 LOG_RESULTS:

5B	00000000G	8F	D0	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	2085
5A	00000000G	00	9E	00009	MOVL	#TERMS TTSET_NUM-1, R11	
59	00000000G	00	9E	00010	MOVAB	TERMS TTSET_KEY, R10	
58	00000000G	00	9E	00017	MOVAB	TERMS TTSET_BIT, R9	
57	00000000G	CF	9E	0001E	MOVAB	SYSSFAOL, R8	
56	00000000G	00	9E	00023	MOVAB	SD COMMA, R7	
5E		3C	C2	0002A	MOVAB	STR\$APPEND, R6	
52	04	AC	D0	0002D	SUBL2	#60, SP	
54	1C	A2	9E	00031	MOVL	DATA BUFFER, R2	2107
55	28	A2	9E	00035	MOVAB	28(R2), R4	
34	AE 020E0000	8F	D0	00039	MOVAB	40(R2), R5	
10	AE	38	AE	D4 00041	MOVL	#34471936, DESC	2117
53		14	AE	9E 00044	CLRL	DESC+4	
		34	A2	D0 00049	MOVAB	FAO BUFFER, FAO_DESC+4	2118
		15	19	0004D	MOVL	52(R2), R3	2123
	00000000G0043	43	DD	0004F	BLSS	1\$	
	38	AE	9F	00056	PUSHL	TERMS_NAME[R3]	2126
66		02	FB	00059	PUSHAB	DESC	
	38	57	DD	0005C	CALLS	#2, STR\$APPEND	
		AE	9F	0005E	PUSHL	R7	2127
66		02	FB	00061	PUSHAB	DESC	
FFFFFFFFFF	8F	53	D1	00064	CALLS	#2, STR\$APPEND	
		11	18	0006B	CMPL	R3, #-1	2134
	44	A2	9F	0006D	BGEQ	2\$	
	38	AE	9F	00070	PUSHAB	68(R2)	2137
66		02	FB	00073	PUSHAB	DESC	
	38	57	DD	00076	CALLS	#2, STR\$APPEND	
		AE	9F	00078	PUSHL	R7	2138
66		02	FB	0007B	PUSHAB	DESC	
53		01	CE	0007E	CALLS	#2, STR\$APPEND	
		17	11	00081	MNEGL	#1, I	2145
6943		62	D3	00083	BRB	4\$	
		11	13	00087	BITL	(R2), TERMS TTSET_BIT[I]	2147
	6A43	DD	DD	00089	BEQL	4\$	
	38	AE	9F	0008C	PUSHL	TERMS TTSET_KEY[I]	2150
66		02	FB	0008F	PUSHAB	DESC	
		57	DD	00092	CALLS	#2, STR\$APPEND	
	38	AE	9F	00094	PUSHL	R7	2151
66		02	FB	00097	PUSHAB	DESC	
E5	53	5B	F3	0009A	CALLS	#2, STR\$APPEND	
	53	03	D0	0009E	AOBLEQ	R11, I, 3\$	2145
		20	11	000A1	MOVL	#3, I	2155
00000000G0043	04	A2	D3	000A3	BRB	6\$	
		15	13	000AC	BITL	4(R2), TERMS TTCLR_BIT[I]	2157
	00000000G0043	43	DD	000AE	BEQL	6\$	
	38	AE	9F	000B5	PUSHL	TERMS TTCLR_KEY[I]	2160
					PUSHAB	DESC	

	66	02	FB	000B8	CALLS	#2, STR\$APPEND		
		57	DD	000BB	PUSHL	R7		2161
	38	AE	9F	000BD	PUSHAB	DESC		
D8	66	02	FB	000C0	CALLS	#2, STR\$APPEND		
53	53	8F	F3	000C3	AOBLEQ	#TERMS_TTCLR_NUM-1, 1, 5\$		2155
53		01	CE	000CB	MNEGL	#1, 1		2165
		20	11	000CE	BRB	8\$		
	00000000G0043	08	A2	D3	BITL	8(R2), TERMS_TT2SET_BIT[1]		2167
		15	13	000D9	BEQL	8\$		
	00000000G0043	DD	000DB	PUSHL	TERMS_TT2SET_KEY[1]			2170
	38	AE	9F	000E2	PUSHAB	DESC		
	66	02	FB	000E5	CALLS	#2, STR\$APPEND		
		57	DD	000E8	PUSHL	R7		2171
	38	AE	9F	000EA	PUSHAB	DESC		
D8	66	02	FB	000ED	CALLS	#2, STR\$APPEND		
53	53	8F	F3	000F0	AOBLEQ	#TERMS_TT2SET_NUM-1, 1, 7\$		2165
53		01	D0	000F8	MOVL	#1, 1		2175
		20	11	000FB	BRB	10\$		
	00000000G0043	0C	A2	D3	BITL	12(R2), TERMS_TT2CLR_BIT[1]		2177
		15	13	00106	BEQL	10\$		
	00000000G0043	DD	00108	PUSHL	TERMS_TT2CLR_KEY[1]			2180
	38	AE	9F	0010F	PUSHAB	DESC		
	66	02	FB	00112	CALLS	#2, STR\$APPEND		
		57	DD	00115	PUSHL	R7		2181
	38	AE	9F	00117	PUSHAB	DESC		
D8	66	02	FB	0011A	CALLS	#2, STR\$APPEND		
53	53	8F	F3	0011D	AOBLEQ	#TERMS_TT2CLR_NUM-1, 1, 9\$		2175
53		01	CE	00125	MNEGL	#1, 1		2185
		21	11	00128	BRB	12\$		
	6943	04	A2	D3	BITL	4(R2), TERMS_TTSET_BIT[1]		2187
		1A	13	0012F	BEQL	12\$		
		D0	A7	9F	PUSHAB	SD NO		2190
	38	AE	9F	00134	PUSHAB	DESC		
	66	02	FB	00137	CALLS	#2, STR\$APPEND		
		6A43	DD	0013A	PUSHL	TERMS_TTSET_KEY[1]		2191
	38	AE	9F	0013D	PUSHAB	DESC		
	66	02	FB	00140	CALLS	#2, STR\$APPEND		
		57	DD	00143	PUSHL	R7		2192
	38	AE	9F	00145	PUSHAB	DESC		
D8	66	02	FB	00148	CALLS	#2, STR\$APPEND		
53	53	5B	F3	0014B	AOBLEQ	R11, 1, 11\$		2185
53		03	D0	0014F	MOVL	#3, 1		2196
		28	11	00152	BRB	14\$		
	00000000G0043	62	D3	00154	BITL	(R2), TERMS_TTCLR_BIT[1]		2198
		1E	13	0015C	BEQL	14\$		
		D0	A7	9F	PUSHAB	SD NO		2201
	38	AE	9F	00161	PUSHAB	DESC		
	66	02	FB	00164	CALLS	#2, STR\$APPEND		
	00000000G0043	DD	00167	PUSHL	TERMS_TTCLR_KEY[1]			2202
	38	AE	9F	0016E	PUSHAB	DESC		
	66	02	FB	00171	CALLS	#2, STR\$APPEND		
		57	DD	00174	PUSHL	R7		2203
	38	AE	9F	00176	PUSHAB	DESC		
D0	66	02	FB	00179	CALLS	#2, STR\$APPEND		
53	53	8F	F3	0017C	AOBLEQ	#TERMS_TTCLR_NUM-1, 1, 13\$		2196
53		01	CE	00184	MNEGL	#1, 1		2207
		29	11	00187	BRB	16\$		

		00000000G0043	0C	A2	D3	00189	15\$:	BITL	12(R2), TERMS_TT2SET_BIT[1]	2209
				1E	13	00192		BEQL	16\$	2212
			D0	A7	9F	00194		PUSHAB	SD_NO	2213
			38	AE	9F	00197		PUSHAB	DESC	2214
	66			02	FB	0019A		CALLS	#2, STR\$APPEND	2215
		00000000G0043		DD	0019D			PUSHL	TERMS_TT2SET_KEY[1]	2216
			38	AE	9F	001A4		PUSHAB	DESC	2217
	66			02	FB	001A7		CALLS	#2, STR\$APPEND	2218
				57	DD	001AA		PUSHL	R7	2219
			38	AE	9F	001AC		PUSHAB	DESC	2220
	66			02	FB	001AF		CALLS	#2, STR\$APPEND	2221
CF	53	00000000G		8F	F3	001B2	16\$:	AOBLEQ	#TERMS_TT2SET_NUM-1, 1, 15\$	2222
	53			01	D0	001BA		MOVL	#1, 1	2223
				29	11	001BD		BRB	18\$	2224
		00000000G0043	08	A2	D3	001BF	17\$:	BITL	8(R2), TERMS_TT2CLR_BIT[1]	2225
				1E	13	001C8		BEQL	18\$	2226
			D0	A7	9F	001CA		PUSHAB	SD_NO	2227
			38	AE	9F	001CD		PUSHAB	DESC	2228
	66			02	FB	001D0		CALLS	#2, STR\$APPEND	2229
		00000000G0043		DD	001D3			PUSHL	TERMS_TT2CLR_KEY[1]	2230
			38	AE	9F	001DA		PUSHAB	DESC	2231
	66			02	FB	001DD		CALLS	#2, STR\$APPEND	2232
				57	DD	001E0		PUSHL	R7	2233
			38	AE	9F	001E2		PUSHAB	DESC	2234
	66			02	FB	001E5		CALLS	#2, STR\$APPEND	2235
CF	53	00000000G		8F	F3	001E8	18\$:	AOBLEQ	#TERMS_TT2CLR_NUM-1, 1, 17\$	2236
11	3C	A2		1D	E1	001F0		BBC	#29, 60(R2), 19\$	2237
			C4	A7	9F	001F5		PUSHAB	SD_DEC_CRT	2238
			38	AE	9F	001F8		PUSHAB	DESC	2239
	66			02	FB	001FB		CALLS	#2, STR\$APPEND	2240
				57	DD	001FE		PUSHL	R7	2241
			38	AE	9F	00200		PUSHAB	DESC	2242
	66			02	FB	00203		CALLS	#2, STR\$APPEND	2243
11	3C	A2		1E	E1	00206	19\$:	BBC	#30, 60(R2), 20\$	2244
			F4	A7	9F	0020B		PUSHAB	SD_DEC_CRT2	2245
			38	AE	9F	0020E		PUSHAB	DESC	2246
	66			02	FB	00211		CALLS	#2, STR\$APPEND	2247
				57	DD	00214		PUSHL	R7	2248
			38	AE	9F	00216		PUSHAB	DESC	2249
	66			02	FB	00219		CALLS	#2, STR\$APPEND	2250
1A	40	A2		1D	E1	0021C	20\$:	BBC	#29, 64(R2), 21\$	2251
			D0	A7	9F	00221		PUSHAB	SD_NO	2252
			38	AE	9F	00224		PUSHAB	DESC	2253
	66			02	FB	00227		CALLS	#2, STR\$APPEND	2254
			C4	A7	9F	0022A		PUSHAB	SD_DEC_CRT	2255
			38	AE	9F	0022D		PUSHAB	DESC	2256
	66			02	FB	00230		CALLS	#2, STR\$APPEND	2257
				57	DD	00233		PUSHL	R7	2258
			38	AE	9F	00235		PUSHAB	DESC	2259
	66			02	FB	00238		CALLS	#2, STR\$APPEND	2260
1A	40	A2		1E	E1	0023B	21\$:	BBC	#30, 64(R2), 22\$	2261
			D0	A7	9F	00240		PUSHAB	SD_NO	2262
			38	AE	9F	00243		PUSHAB	DESC	2263
	66			02	FB	00246		CALLS	#2, STR\$APPEND	2264
			F4	A7	9F	00249		PUSHAB	SD_DEC_CRT2	2265
			38	AE	9F	0024C		PUSHAB	DESC	2266
	66			02	FB	0024F		CALLS	#2, STR\$APPEND	2267

				57	DD	00252	PUSHL	R7		2257
				AE	9F	00254	PUSHAB	DESC		
		66		02	FB	00257	CALLS	#2, STR\$APPEND		
06		64		04	E1	0025A	BBC	#4, (R4), 23\$		2262
			0204	C7	9F	0025E	PUSHAB	P.ACQ		2265
				2E	11	00262	BRB	26\$		
04		64		02	E0	00264	BBS	#2, (R4), 24\$		2268
34		64		03	E1	00268	BBC	#3, (R4), 27\$		
			FF58	C7	9F	0026C	PUSHAB	SD PARITY		2271
			38	AE	9F	00270	PUSHAB	DESC		
		66		02	FB	00273	CALLS	#2, STR\$APPEND		
			0210	C7	9F	00276	PUSHAB	P.ACS		2272
			38	AE	9F	0027A	PUSHAB	DESC		
08		66		02	FB	0027D	CALLS	#2, STR\$APPEND		
		64		02	E1	00280	BBC	#2, (R4), 25\$		2273
			00000000G	00	9F	00284	PUSHAB	TERMS_ODD		2274
				06	11	0028A	BRB	26\$		
			00000000G	00	9F	0028C	PUSHAB	TERMS_EVEN		2275
			38	AE	9F	00292	PUSHAB	DESC		
		66		02	FB	00295	CALLS	#2, STR\$APPEND		
				57	DD	00298	PUSHL	R7		2276
			38	AE	9F	0029A	PUSHAB	DESC		
		66		02	FB	0029D	CALLS	#2, STR\$APPEND		
20		64		06	E1	002A0	BBC	#6, (R4), 28\$		2282
	OC	AE		20	D0	002A4	MOVL	#32, FAO_DESC		2285
		6E		A2	9A	002A8	MOVZBL	24(R2), (SP)		2289
			18	5E	DD	002AC	PUSHL	SP		
			10	AE	9F	002AE	PUSHAB	FAO_DESC		
			14	AE	9F	002B1	PUSHAB	FAO_DESC		
			0224	C7	9F	002B4	PUSHAB	P.ACQ		
		68		04	FB	002B8	CALLS	#4, SYSSFAOL		
			OC	AE	9F	002BB	PUSHAB	FAO_DESC		2290
			38	AE	9F	002BE	PUSHAB	DESC		
		66		02	FB	002C1	CALLS	#2, STR\$APPEND		
22		64		0A	E1	002C4	BBC	#10, (R4), 29\$		2295
	OC	AE		20	D0	002C8	MOVL	#32, FAO_DESC		2298
6E	14	A2		00	EF	002CC	EXTZV	#0, #4, 20(R2), (SP)		2302
				5E	DD	002D2	PUSHL	SP		
			10	AE	9F	002D4	PUSHAB	FAO_DESC		
			14	AE	9F	002D7	PUSHAB	FAO_DESC		
			0238	C7	9F	002DA	PUSHAB	P.ACQ		
		68		04	FB	002DE	CALLS	#4, SYSSFAOL		
			OC	AE	9F	002E1	PUSHAB	FAO_DESC		2303
			38	AE	9F	002E4	PUSHAB	DESC		
		66		02	FB	002E7	CALLS	#2, STR\$APPEND		
06		64		0B	E1	002EA	BBC	#11, (R4), 30\$		2308
			0258	C7	9F	002EE	PUSHAB	P.ACQ		2311
				08	11	002F2	BRB	31\$		
12		64		OC	E1	002F4	BBC	#12, (R4), 32\$		2314
			0278	C7	9F	002F8	PUSHAB	P.ADA		2317
			38	AE	9F	002FC	PUSHAB	DESC		
		66		02	FB	002FF	CALLS	#2, STR\$APPEND		
				57	DD	00302	PUSHL	R7		2318
			38	AE	9F	00304	PUSHAB	DESC		
		66		02	FB	00307	CALLS	#2, STR\$APPEND		
20		64		05	E1	0030A	BBC	#5, (R4), 33\$		2323
	OC	AE		20	D0	0030E	MOVL	#32, FAO_DESC		2326

20	6E	19	A2	9A	00312	MOVZBL	25(R2), (SP)	2330	
		10	5E	DD	00316	PUSHL	SP		
		14	AE	9F	00318	PUSHAB	FAO_DESC		
		028C	AE	9F	0031B	PUSHAB	FAO_DESC		
			C7	9F	0031E	PUSHAB	P.ADC		
	68		04	FB	00322	CALLS	#4, SYSSFAOL		
		0C	AE	9F	00325	PUSHAB	FAO_DESC	2331	
		38	AE	9F	00328	PUSHAB	DESC		
	66		02	FB	0032B	CALLS	#2, STR\$APPEND		
	64		09	E1	0032E	BBC	#9, (R4), 34\$	2337	
OC	AE		20	DO	00332	MOVL	#32, FAO_DESC	2340	
	6E	07	A5	9A	00336	MOVZBL	7(R5), (SP)	2344	
			5E	DD	0033A	PUSHL	SP		
		10	AE	9F	0033C	PUSHAB	FAO_DESC		
		14	AE	9F	0033F	PUSHAB	FAO_DESC		
		02A0	C7	9F	00342	PUSHAB	P.ADE		
	68		04	FB	00346	CALLS	#4, SYSSFAOL		
		0C	AE	9F	00349	PUSHAB	FAO_DESC	2345	
		38	AE	9F	0034C	PUSHAB	DESC		
	66		02	FB	0034F	CALLS	#2, STR\$APPEND		
OC	20	01	A4	E9	00352	BLBC	1(R4), 35\$	2351	
	AE		20	DO	00356	MOVL	#32, FAO_DESC	2354	
	6E	02	A5	3C	0035A	MOVZWL	2(R5), (SP)	2358	
			5E	DD	0035E	PUSHL	SP		
		10	AE	9F	00360	PUSHAB	FAO_DESC		
		14	AE	9F	00363	PUSHAB	FAO_DESC		
		02B4	C7	9F	00366	PUSHAB	P.ADG		
	68		04	FB	0036A	CALLS	#4, SYSSFAOL		
		0C	AE	9F	0036D	PUSHAB	FAO_DESC	2359	
		38	AE	9F	00370	PUSHAB	DESC		
	66		02	FB	00373	CALLS	#2, STR\$APPEND		
			64	95	00376	TSTB	(R4)	2365	
			40	18	00378	BGEQ	38\$		
OC	AE		20	DO	0037A	MOVL	#32, FAO_DESC	2370	
	50	10	A2	9A	0037E	MOVZBL	16(R2), R0	2371	
04	AE	00000000G00	40	DO	00382	MOVL	TERMS_SPDBLK[R0], SPEEDS		
	50	11	A2	9A	0038B	MOVZBL	17(R2), R0	2372	
			0B	13	0038F	BEQL	36\$		
08	AE	00000000G00	40	DO	00391	MOVL	TERMS_SPDBLK[R0], SPEEDS+4	2373	
			05	11	0039A	BRB	37\$		
08	AE		04	AE	DO	0039C	MOVL	SPEEDS, SPEEDS+4	2374
			04	AE	9F	003A1	PUSHAB	SPEEDS	2378
			10	AE	9F	003A4	PUSHAB	FAO_DESC	
			14	AE	9F	003A7	PUSHAB	FAO_DESC	
		02CC	C7	9F	003AA	PUSHAB	P.ADI		
	68		04	FB	003AE	CALLS	#4, SYSSFAOL		
		0C	AE	9F	003B1	PUSHAB	FAO_DESC	2379	
		38	AE	9F	003B4	PUSHAB	DESC		
	66		02	FB	003B7	CALLS	#2, STR\$APPEND		
		34	AE	B5	003BA	TSTW	DESC	2385	
			18	13	003BD	BEQL	39\$		
		34	AE	B7	003BF	DECW	DESC	2388	
		34	AE	9F	003C2	PUSHAB	DESC	2389	
		20	A2	9F	003C5	PUSHAB	32(R2)		
			02	DD	003C8	PUSHL	#2		
00000000G	00	00000000G	8F	DD	003CA	PUSHL	#SETS_TERMSET		
			04	FB	003D0	CALLS	#4, LIB\$SIGNAL		

SETTERM
V04-000

L 15
16-Sep-1984 01:10:06
14-Sep-1984 12:09:20

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETTERM.B32;1

Page 77
(12)

04 003D7 39\$: RET

; 2392

: Routine Size: 984 bytes, Routine Base: \$CODE\$ + 102B

SETTERM
V04-000

M 15
16-Sep-1984 01:10:06
14-Sep-1984 12:09:20

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SETTERM.B32;1

Page 78
(13)

: 2307
: 2308
2393 1 END
2394 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	236	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	932	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	5123	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	120	0	1000	00:02.0

: Information: 1
: Warnings: 0
: Errors: 0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:SETTERM/OBJ=OBJ\$:SETTERM MSRC\$:SETTERM/UPDATE=(ENH\$:SETTERM)

: Size: 5123 code + 1168 data bytes
: Run Time: 01:36.3
: Elapsed Time: 05:01.3
: Lines/CPU Min: 1491
: Lexemes/CPU-Min: 21433
: Memory Used: 464 pages
: Compilation Complete

0054 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SETPROCES
LIS

SETSHOBRO
LIS

SETVOLUME
LIS

SETPWD
LIS

SETTERM
LIS

SETQUEUE
LIS

SETTIME
LIS